## Contents


### PART 1  NLS Concepts 1

**Chapter 1 • National Language Support (NLS)** .................................. 3
  Overview to National Language Support .............................................. 3
  Definition of Localization and Internationalization .............................. 4

**Chapter 2 • Locale for NLS** ........................................................... 5
  Overview of Locale Concepts for NLS .................................................. 5
  Specifying a Locale ............................................................................. 6

**Chapter 3 • Encoding for NLS** ......................................................... 9
  Overview: Encoding for NLS ............................................................. 9
  Difference between Encoding and Transcoding ..................................... 12
  Character Sets for Encoding in NLS ..................................................... 12
  Common Encoding Methods ............................................................... 13
  Standards Organizations for NLS Encodings ....................................... 15
  Code Point Discrepancies among EBCDIC Encodings ........................... 16
  Collating Sequence ............................................................................. 17
  Determining the Encoding of a SAS Session and a Data Set ................... 21
  Default SAS Session Encoding .......................................................... 22
  Setting the Encoding of a SAS Session ................................................. 23
  Encoding Behavior in a SAS Session ................................................... 25

**Chapter 4 • Transcoding for NLS** ..................................................... 27
  Overview to Transcoding ................................................................... 27
  Common Reasons for Transcoding ...................................................... 28
  Transcoding and Translation Tables ..................................................... 28
  SAS Options That Transcode SAS Data ................................................. 30
  Transcoding between Operating Environments ..................................... 30
  Transcoding Considerations ............................................................... 31
  Compatible and Incompatible Encodings .............................................. 32
  Preventing Transcoding ..................................................................... 33
  Avoiding Character Data Truncation By Using the CVP Engine ............... 34

**Chapter 5 • Double-Byte Character Sets (DBCS)** ................................. 37
  Overview to Double-Byte Character Sets (DBCS) .................................. 37
  East Asian Languages ......................................................................... 38
  Specifying DBCS ............................................................................... 38
  Requirements for Displaying DBCS Character Sets ............................... 38
  When You Can Use DBCS Features ..................................................... 39
  DBCS and SAS on a Mainframe ............................................................ 39
  SAS Data Conversion between DBCS Encodings .................................... 40
  Avoiding Problems with Split DBCS Character Strings ........................... 40
PART 7  Macro Functions for NLS  563

Chapter 14 • Dictionary of Macro Functions for NLS ............................................. 565
  Macro Functions by Category ............................................. 565
  Dictionary ............................................. 566

PART 8  System Options for NLS  575

Chapter 15 • Dictionary of System Options for NLS ............................................. 577
  System Option Entries by Category ............................................. 577
  Dictionary ............................................. 579

PART 9  Options for Commands, Statements, and Procedures for NLS  613

Chapter 16 • Dictionary of Command, Statement, and Procedure Option for NLS ........ 615
  Commands, Statements, and Procedures for NLS by Category ............................................. 615
  Dictionary ............................................. 616

PART 10  Procedures for NLS  647

Chapter 17 • DBCSTAB Procedure ............................................. 649
  Overview: DBCSTAB Procedure ............................................. 649
  Syntax: DBCSTAB Procedure ............................................. 649
  Examples: DBCSTAB Procedure ............................................. 651

Chapter 18 • LOCALEDATA Procedure ............................................. 655
  Overview: LOCALEDATA Procedure ............................................. 655
  Syntax: LOCALEDATA Procedure ............................................. 655
  Example: Modifying Locale Data Using the LOCALEDATA Procedure ............................................. 663

Chapter 19 • TRANTAB Procedure ............................................. 669
  Overview: TRANTAB Procedure ............................................. 669
  Concepts: TRANTAB Procedure ............................................. 670
  Syntax: TRANTAB Procedure ............................................. 672
  Examples: TRANTAB Procedure ............................................. 678

PART 11  Values for Locale, Encoding, and Transcoding  695

Chapter 20 • Values for the LOCALE= System Option ............................................. 697
  LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options ............................................. 697

Chapter 21 • SAS System Options for Processing DBCS Data ............................................. 711
  Overview to System Options Used in a SAS Session for DBCS ............................................. 711
What's New in SAS 9.4 National Language Support

Overview

This release expands the scope and capabilities of SAS National Language Support (NLS). NLS is a set of features that enables a software product to function properly in every global market for which the product is targeted. SAS contains NLS features to ensure that you can write SAS applications that conform to local language conventions. Typically, software that is written in the English language works well for users who speak English and for data that is formatted using the conventions that are observed in the United States. However, without NLS, these products might not work as well for users in other regions of the world. SAS NLS enables users in regions such as Asia and Europe to process data successfully in their native languages and environments.

General Enhancements

The following enhancements are implemented for SAS 9.4 National Language Support:

- “Exceptions for Date and Time Default Widths” on page 79 provides exceptions to default widths. Some format widths exceed the default width, depending on the locale and encoding.

- The SAS name and POSIX name, Serbian_Yugoslavia sr_YU, has been removed from the document.

- “Specifying Time Zones in SAS” on page 41 explains SAS time zones.

- Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819 lists time zone IDs and time zone names.

For the second maintenance release, the following chapter was added. Time zone information was taken from the overview and added to this chapter.

- Chapter 6, “Time Zones,” on page 41

Additional Encodings

The following encodings are new:
Locales

The following locales were added:

- English_Malta (en_MT)
- Greek_Cyprus (el_CY)
- Irish_Ireland (ga_IE)

For more information, see the LOCALE= table on page 697.

For the third maintenance release, the ManxGaelic_UnitedKingdom (gv_GB) locale was deprecated.

For the third maintenance release, the following locales are new:

- Kazakh_Kazakhstan (kk_KZ)
- Basque_Spain (eu_ES)

Formats

The following formats are new:

**B8601DX** (p. 99)
Converts UTC datetime values into user local time and writes local time (SAS datetime values) with time zone offsets.

**B8601LX** (p. 101)
Writes SAS datetime values with time zone offsets.

**B8601TX** (p. 102)
Converts UTC time into user local time and writes local time values with time zone offsets.

**E8601DX** (p. 110)
Converts UTC datetime values into user local time and writes local time (SAS datetime values) with time zone offsets.

**E8601LX** (p. 111)
Writes SAS datetime values with time zone offsets.

**E8601TX** (p. 113)
Converts UTC time into user local time and writes local time values with time zone offsets.

**NLDATEL** (p. 132)
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date in the form month, date, year.
NLDATEM (p. 132)
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date.

NLDATEMDL (p. 134)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.

NLDATEMDM (p. 135)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.

NLDATEMDS (p. 136)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.

NLDATES (p. 138)
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date string.

NLDATEMYML (p. 142)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and year.

NLDATEMYMM (p. 143)
Converts a SAS date value to the date string of the specified locale and then writes the date values as the month and year with abbreviations.

NLDATEMYMS (p. 144)
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date and year.

NLDATEMYQL (p. 146)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.

NLDATEMYQM (p. 147)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.

NLDATEMYQS (p. 148)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value (1–4) with numbers and delimiters.

NLDATML (p. 154)
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date in the form month, date, year, and time.

NLDATMM (p. 155)
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date and time with abbreviations for the month and time.

NLDATMMDL (p. 156)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month.

NLDATMYQS (p. 172)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the year’s quarter value using numbers and delimiters.
NLDATMMDM (p. 157)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month using abbreviations.

NLDATMMDS (p. 158)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and day of the month using numbers and delimiters.

NLDATMS (p. 160)
Converts a SAS date value to the date string of the specified locale and then writes the date value as a date in the form MM/DD/YYYY.

NLDATMYML (p. 167)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and the year.

NLDATMYMM (p. 168)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the month and the year.

NLDATMYQM (p. 171)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year’s quarter value (1–4) and the year.

NLDATMYQS (p. 172)
Converts a SAS date value to the date string of the specified locale and then writes the date value as the year and the quarter (1-4) using numbers and delimiters.

NLDATMYMS (p. 168)
Converts a SAS date value to the date string of the specified locale and then writes the month and year with numbers and delimiters.

For the second maintenance release, the following format is new:

BESTDOTX (p. 104)
Specifies that SAS choose the best notation and use a dot as a decimal separator.

For the second maintenance release, the country Latvia was added to the following formats:

NLMNIEUR (p. 187)
Writes the monetary format of the international expression for Belgium, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

NLMNLEUR (p. 224)
Writes the monetary format of the local expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

For the third maintenance release, the following formats were modified:

NLDATEYM (p. 141)
Includes a note and example explaining how to use PROC LOCALEDATA to process data with a width of 6.

NLMNIEUR (p. 187)
Writes the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.
NLNMLEUR (p. 224)

Writes the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

Functions

The following functions are new:

**BASECHAR** (p. 334)
Converts characters to base characters.

**KSTRIP** (p. 364)
Removes leading and trailing blanks from a character string.

**SETLOCALE** (p. 395)
Specifies the locale keys for the current SAS locale.

**TZONEID** (p. 402)
Returns the current time zone ID.

**TZONENAME** (p. 404)
Returns the current standard or daylight saving time time zone name.

**TZONES2U** (p. 406)
Converts a SAS datetime value to a UTC datetime value.

**TZONEU2S** (p. 411)
Converts a UTC datetime value to a SAS datetime value.

An example was added in the following functions:

- “KCOMPARE Function” on page 342
- “KCOMPRESS Function” on page 343
- “KCOUNT Function” on page 344
- “KINDEX Function” on page 347
- “KINDEXC Function” on page 350
- “KLEFT Function” on page 351
- “KLENGTH Function” on page 352
- “KLOWCASE Function” on page 353
- “KREVERSE Function” on page 360
- “KRIGHT Function” on page 361
- “KSCAN Function” on page 362
- “KSTRCAT Function” on page 363
- “KSUBSTR Function” on page 366
- “KSUBSTRB Function” on page 367
- “KTRANSLATE Function” on page 368
- “KTRIM Function” on page 370
- “KTRUNCATE Function” on page 371
For the first maintenance release, the following function is new:

ANORM420 (p. 330)
Returns a normalized string from an input string encoded in EBCDIC420.

For the second maintenance release, the following functions are new:

KUPDATES (p. 376)
Inserts, deletes, and replaces character value contents.
TZONEDSTNAME (p. 407)
Returns a daylight saving time name.
TZONEDSTOFF (p. 408)
Returns the time zone offset value for the specified daylight saving time.
TZONESTTNAME (p. 409)
Returns a standard time zone name.
TZONESTTOFF (p. 410)
Returns the time zone offset value for the specified standard time.

For the second maintenance release, the following functions were updated:

KCVT (p. 345)
DBCS, SBCS, and MBCS information was added and references to DBCSLANG and DBCSTYPE were removed.
KUPDATE (p. 373)
The NLSCOMPATMODE option was removed from the function.

For the third maintenance release, the following functions are new:

KINDEXB (p. 348)
Searches a character expression for a string of characters.
KINDEXCB (p. 349)
Searches a character expression for specified characters.
KVERIFYB (p. 379)
Returns the position of the first character that is unique to an expression.

For the third maintenance release, the I18N level changed for the following functions:

ANORM420 (p. 330)
I18NL1
BASECHAR (p. 334)
I18NL2
GETLOCENV (p. 338)
I18NL2
GETPXLANGUAGE (p. 339)
I18NL2
GETPXLLOCALE (p. 340)
I18NL2
GETPXREGION (p. 341)  I18NL2
KCOMPARE (p. 342)  I18NL2
KCOMPRESS (p. 343)  I18NL2
KCOUNT (p. 344)  I18NL2
KCVT (p. 345)  I18NL1
KINDEX (p. 347)  I18NL2
KINDEXC (p. 350)  I18NL2
KLEFT (p. 351)  I18NL2
KLENGTH (p. 352)  I18NL2
KLOWCASE (p. 353)  I18NL2
KPROPCASE (p. 354)  I18NL2
KPROPCHAR (p. 357)  I18NL2
KPROPDATA (p. 358)  I18NL2
KREVERSE (p. 360)  I18NL2
KRIGHT (p. 361)  I18NL2
KSCAN (p. 362)  I18NL2
KSTRCAT (p. 363)  I18NL2
KSTRIP (p. 364)  I18NL2
KSUBSTR (p. 366)  I18NL2
KSUBSTRB (p. 367)  I18NL2
KTRANSLATE (p. 368)  I18NL2
KTRIM (p. 370)  I18NL2
KTRUNCATE (p. 371)
I18NL2
KUPCASE (p. 372)
I18NL2
KUPDATE (p. 373)
I18NL2
KUPDATEB (p. 375)
I18NL2
KUPDATES (p. 376)
I18NL2
KVERIFY (p. 378)
I18NL2
NLDATE (p. 380)
I18NL2
NLDATM (p. 383)
I18NL2
NLTIME (p. 386)
I18NL2
SASMSG (p. 387)
I18NL2
SASMSGL (p. 390)
I18NL2
SETLOCALE (p. 395)
I18NL2
SORTKEY (p. 392)
I18NL2
TZONEDSTOFF (p. 408)
I18NL2
TZONEDSTNAME (p. 407)
I18NL2
TRANTAB (p. 412)
I18NL2
TZONEID (p. 402)
I18NL2
TZONENAME (p. 404)
I18NL2
TZONEOFF (p. 405)
I18NL2
TZONESTOFF (p. 410)
I18NL2
TZONES2U (p. 406)
I18NL2
TZONESTTNAME (p. 409)
I18NL2
For the third maintenance release, SAS supports the Unicode supplementary character sets with a Unicode session. The following functions were updated:

- “UNICODE Function” on page 413
- “UNICODEC Function” on page 414
- “UNICODEWIDTH Function” on page 416

Informats

For the second maintenance release, the country Latvia was added to the following informats:

NLMNIEUR (p. 463)
Reads the monetary format of the international expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

NLMNLEUR (p. 500)
Reads the monetary format of the local expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

For the second maintenance release, the following informats are new:

NLDATEW (p. 448)
Reads the date value in the specified locale and then converts the date value to the local SAS date and the day of the week.

NLDATMW (p. 451)
Reads the date value in the specified locale and then converts the date value to the local SAS day of the week and the datetime.

NLDATMAP (p. 450)
Reads the date value in the specified locale and then converts the date value to the local SAS datetime with either a.m. or p.m.

For the second maintenance release, aliases were added to the following informats:

NLDATE (p. 447)
Reads the date value in the specified locale and then converts the date value to the local SAS date value.
NLTIME (p. 535)
Reads the time value in the specified locale and then converts the time value to the local SAS time value.

NLDATM (p. 449)
Reads the datetime value of the specified locale and then converts the datetime value to the local SAS datetime value.

For the third maintenance release, the following informats were updated:

NLMNIEUR (p. 463)
Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

NLMNLEUR (p. 500)
Reads the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

---

Procedures

The following procedure is new:

LOCALEDATA (p. 655)
Enables you to customize locale data.

---

System Options

The following system options are new:

LSWLANG (p. 596)
Specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation.

LOGLANGENG (p. 595)
Specifies using the English language for SAS log message text when the locale option is set after start up.

MAPEBCDIC2ASCII= (p. 597)
Specifies a translation table that is used by SAS users to transcode from EBCDIC to ASCII and from ASCII to EBCDIC.

ODSLANGCHG (p. 599)
Specifies whether the language of the text of the ODS output can be changed.

LOCALEDATA (p. 593)
Specifies the source database for the locale information.

TIMEZONE (p. 604)
Specifies the user local time zone.

For the second maintenance release, the following system option is new:
NLDECSEPARATOR (p. 598)
   Specifies whether SAS produces locale-sensitive numeric output for the decimal
   separator or continues to format numbers with U.S. English preferences.

The following system option was enhanced:

DFLANG (p. 585)
   Supports the locale option.

For the second maintenance release, the NLSCOMPATMODE system option was
removed.

For the third maintenance release, the LINGUISTIC option was added to the
SORTSEQ= system option on page 602.
SAS National Language Support
### Part 1

**NLS Concepts**

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Language Support (NLS)</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Locale for NLS</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Encoding for NLS</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Transcoding for NLS</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>Double-Byte Character Sets (DBCS)</td>
<td>37</td>
</tr>
<tr>
<td>6</td>
<td>Time Zones</td>
<td>41</td>
</tr>
</tbody>
</table>
Chapter 1
National Language Support (NLS)

Overview to National Language Support

Introduction to National Language Support

Definition of Localization and Internationalization

Overview to National Language Support

Overview to National Language Support

National Language Support (NLS) is a set of features that enable a software product to function properly in every global market for which the product is targeted. The SAS System contains NLS features to ensure that SAS applications can be written so that they conform to local language conventions. Typically, software that is written in the English language works well for users who use the English language and use data that is formatted using the conventions that are observed in the United States. However, without NLS, these products might not work well for users in other regions of the world.

NLS in SAS enables users in regions such as Asia and Europe to process data successfully in their native languages and environments.

SAS provides NLS for data as well as for code under all operating environments and hardware, from the mainframe to the personal computer. This support is especially important to international users who are running applications in a client/server environment. SAS provides NLS for mainframes while maintaining consistency with applications that were developed with previous versions of SAS.

NLS is applied to data that is moved between machines; for example, NLS ensures that the data is converted to the correct format for use on the target machine.

Text-string operations are sensitive to SAS settings for language and region. This action enables correct results for such operations as uppercasing and lowercasing characters, classifying characters, and scanning data. SAS provides features to ensure that national characters, which are characters specific to a particular nation or group of nations, are displayed and are printed properly.

Software applications that incorporate NLS can avoid dependencies on language-specific or cultural-specific conventions for software features such as:

- character classifications
- character comparison rules
- code sets
- date and time formatting
- User interface
Definition of Localization and Internationalization

*Internationalization* is the process of designing a software product without making assumptions that are based on a single language or locale, thereby facilitating localization. Internationalization ensures that international conventions (including rules for sorting strings and for formatting dates, times, numbers, and currencies) are supported. It also facilitates a consistent user experience across different language editions of a product. The abbreviation for internationalization is I10N.

Although the application logic might support cultural conventions (for example, the monetary and numeric formats of a particular region), only a localized version of the software presents user interfaces and system messages in the local language.

*Localization* is the process of adapting a product to meet the language, cultural, and other requirements of a specific target environment or market so that users can use their own languages and conventions when using the product. Translation of the user interface, system messages, and documentation is part of localization.

SAS NLS features are available for localizing and internationalizing your SAS applications. For more information see “Internationalization Compatibility for SAS String Functions” on page 307.
Overview of Locale Concepts for NLS

A locale reflects the language, local conventions such as data formatting, and culture for a geographical region. Local conventions might include specific formatting rules for dates, times, and numbers and a currency symbol for the country or region. Collating sequence, paper size, postal addresses, and telephone numbers can also be included in locale.

Dates have many representations, depending on the conventions that are accepted in a culture. The month might be represented as a number or as a name. The name might be fully spelled or abbreviated. The order of the month, day, and year might differ according to locale.

For example, “the third day of October in the year 2002” would be displayed in a different way for each of these locales:

Canada
02–10–03

Germany
03.10.02

Italy
3/10/02

United States
10/03/02

Time can be represented in one English-speaking country or region by using the 12-hour notation. Other English speakers expect time values to be formatted using the 24-hour notation.

Language is part of a locale, but is not unique to any one locale. For example, Portuguese is spoken in Brazil as well as in Portugal, but the cultures are different. In Brazil and in Portugal, there are similarities in the formatting of data. Numbers are
formatted using a comma (,) to separate integers from fractional values and a dot (.) to separate groups of digits to the left of the radix character. However, there are important differences, such as the currency symbols that are used in the two different locales.

Portugal uses the Euro and requires the Euro symbol € while Brazil uses the Real that is represented by the two-character currency symbol R$. Also, a country might have more than one official language. Canada has two official languages: English and French; two values can be specified for the LOCALE= system option: English_Canada and French_Canada.

Numbers, including currency, can have different representations. For example, the decimal separator, or radix character, is a dot (.) in some regions and a comma (,) in other regions. The thousands separator can be a dot, comma, or even a space in some regions. Monetary conventions likewise vary between locales; for example, a dollar sign or a yen sign might be attached to a monetary value.

Paper size and measurement are also locale considerations. Standard paper sizes include letter (8-1/2-by-11-inch paper) and A4 (210-by-297-millimeter paper). The letter paper size is mainly used by some English-speaking countries; A4 is used by most other locales. Most locales use centimeters, some locales use inches.

Specify a Locale

How Locale Is Specified at SAS Invocation

You can use the LOCALE= system option to specify the locale of the SAS session at SAS invocation. LOCALE= also implicitly sets the following SAS system options:

- DATESTYLE=
- DFLANG=
- ENCODING=
- LOCALEDATA
- MAPEBCDICTOASCII
- ODSLANGCHG
- PAPERSIZE=
- RSASIOTRANSERROR
- TIMEZONE
- TRANTAB=
- URLENCODING

Windows example:

```bash
sas9 -locale English_UnitedStates
```

Note: Locale can also be specified using POSIX naming standards. For example, en_US is the POSIX equivalent for the SAS value English_UnitedStates.

Default values for the LOCALE= option are the same under each operating environment. For details, see “LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options” on page 697.
The English_UnitedStates value for LOCALE= causes the following options to be implicitly set to the specified default values SAS invocation:

- DATESTYLE=MDY
- DFLANG=English
- ENCODING=wlatin1
- PAPERSIZE=Letter
- TRANTAB=(lat1lat1, lat1lat1, wlt1_ucs, wlt1_lcs, wlt1_ccl, )

At invocation, an explicitly set system option overrides any implicitly set option.

Windows example:
```
sas9 -papersize A4;
```

At invocation, the explicit setting PAPERSIZE=A4 overrides an implicit setting of the PAPERSIZE= option via the LOCALE= option. For details, see “PAGESIZE= System Option” on page 600.

### How Locale Is Specified during a SAS Session

You can use the LOCALE= system option to specify the locale of the SAS session during the SAS session. However, only the values for these system options change implicitly to reflect the changed value of LOCALE=:

- DATESTYLE=
- DFLANG=
- PAPERSIZE=

**Note:** The DATESTYLE and PAPERSIZE options are affected only if these options are set to LOCALE. For more information see “DATESTYLE= System Option” on page 580 and “PAGESIZE= System Option” on page 600.

The values for these system options do not change implicitly to reflect the changed value of LOCALE=:

- ENCODING=
- TRANTAB=

**Note:** ENCODING= cannot be reset during a SAS session. It can be set only at invocation.

**Note:** For more details about the differences between the LOCALE= and ENCODING= options, see “Setting the Encoding of a SAS Session” on page 23.

Windows example:
```
options locale=Italian_Italy;
```

The Italian_Italy value that is assigned to the LOCALE= option causes the following options to be implicitly reset during the SAS session to reflect the changed value of the LOCALE= system option:

- DATESTYLE=DMY
- DFLANG=Italian
- PAPERSIZE=A4

The values for the ENCODING= and TRANTAB= options do not reset; their former values are retained.
For details about these system options, see “DATESTYLE= System Option” on page 580.

**Language Switching**

SAS messages are displayed in the language that is specified by the settings in the SAS configuration file during start-up. In a SAS session encoding, you can view SAS messages in another language by using the Language Switching feature. You can access the Language Switching feature with the ODSLANGCHG system option. If ODSLANGCHG is enabled, then the value of the LOCALE system option determines the language for procedure output, user interface elements, and ODS fonts. If ODSLANGCHG is disabled, then messages appear in the language that is set during start-up. This feature is supported in the Unicode server. For more information, see the “ODSLANGCHG System Option” on page 599.

The LOGLANGCHG system option controls whether language switching occurs for the SAS log messages. This option controls the language of message switching in SAS log output. If LOGLANGCHG is specified, the language of the SAS log depends on the LSWLANG or LOCALE= option. LSWLANG has the higher priority. If the LSWLANG option is set to a valid SAS language, the SAS log output is controlled by its value. Otherwise, the LOCALE= option determines the language of the SAS log. For more information, see “LOGLANGCHG System Option” on page 594.

The LSWLANG option specifies the language of messages if LOGLANGCHG or ODSLANGCHG are enabled. If LSWLANG is set to LOCALE, the LOCALE= option determines the language for switching. LSWLANG=LOCALE is the default. The setting of the LSWLANG= option also needs to be compatible with SAS session encoding. Otherwise, its value is ignored. For more information, see “LSWLANG System Option” on page 596.

The LOGLANGENG option is a toggle option that overrides LOGLANGCHG and LSWLANG and sets them to LOGLANGCHG=ON and LSWLANG=EN. As a result, the LOG output is in English, and the NL format output has no change. This option changes the setting of ODSLANGCHG. If ODSLANGCHG=OFF, the system message language for ODS output is determined by SAS configuration. If ODSLANGCHG=ON, all messages are in English because of the LSWLANG setting. For more information, see “LOGLANGENG System Option” on page 595.

For more information about Language Switching, see Multilingual Computing with SAS® 9.4.
Chapter 3
Encoding for NLS

Overview: Encoding for NLS

An encoding maps each character in a character set to a unique numeric representation, which results in a table of all code points. This table is referred to as a code page, which is an ordered set of characters in which a numeric index (code point value) is associated with each character. The position of a character on the code page determines its two-digit hexadecimal number.

For example, the following is the code page for the Windows Latin1 encoding. In the following example, the row determines the first digit and the column determines the
second digit. The numeric representation for the uppercase A is the hexadecimal number 41, and the numeric representation for the equal sign (=) is the hexadecimal number 3D.

**Figure 3.1** Windows Latin1 Code Page

A character set is the set of characters and symbols that are used by a language or group of languages. A character set includes national characters (which are characters specific to a particular nation or group of nations), special characters (such as punctuation marks), the unaccented Latin characters A–Z, the digits 0–9, and control characters that are needed by the computer.

An encoding method is a set of rules that assign the numeric representations to the set of characters. These rules govern the size of the encoding (number of bits used to store the numeric representation of the character) and the ranges in the code page where characters appear. The encoding methods result from the adherence to standards that have been developed in the computing industry. An encoding method is often specific to the computer hardware vendor.
An encoding results from applying an encoding method to a character set.

An individual character can occupy a different position in a code page, depending on the code page used. An example is the German uppercase letter Ä:

- is represented as the hexadecimal number C4 in the Windows Latin1 code page (1252)
- is represented as the hexadecimal number 4A in the German EBCDIC code page (1141)

In the following example, the column determines the first digit and the row determines the second digit.

*Figure 3.2* German EBCDIC Code Page

Each SAS session is set to a default encoding, which can be specified by using various SAS language elements.
Difference between Encoding and Transcoding

Encoding establishes the default working environment for your SAS session. For example, the Windows Latin1 encoding is the default encoding for a SAS session under Windows in a Western European locale such as the de_DE locale for German in Germany. For example, the Windows Latin1 code point for the uppercase letter Ä is C4 hexadecimal.

Note: The default encoding varies according to the operating environment and the locale.

However, if you are working in an international environment (for example, you access SAS data that is encoded in German EBCDIC), the German EBCDIC code point for the uppercase letter Ä is 4A hexadecimal. In order for a version of SAS that normally uses Windows Latin1 to properly interpret a data set that is encoded in German EBCDIC, the data must be transcoded. Transcoding is the process of converting data from one encoding to another. When SAS transcodes the Windows Latin1 uppercase letter Ä to the German EBCDIC uppercase letter Ä, the hexadecimal representation for the character is converted from the value C4 to a 4A. For conceptual information, see Chapter 4, “Transcoding for NLS,” on page 27.

Character Sets for Encoding in NLS

Encodings are available to address the requirements of the character set (few languages use the same 26 characters, A through Z as English). All languages are represented using either of the following classes of character sets:

SBCS (Single-Byte Character Set)
represents each character in a single (one) byte. A single-byte character set can be either 7 bits (providing up to 128 characters) or 8 bits (providing up to 256 characters). An example of an 8-bit SBCS is the ISO 8859-5 (Cyrillic) character set (represents the Russian characters).

For details about how SAS uses SBCS encodings, see Chapter 22, “Encoding Values in SAS Language Elements,” on page 713.

DBCS (Double-Byte Character Set)
refers to the East Asian character sets (Japanese, Korean, Simplified Chinese, and Traditional Chinese), which require a mixed-width encoding because most characters consist of more than one byte. Although the term DBCS (Double-Byte Character Set) is more commonly used than MBCS (Multi-Byte Character Set), MBCS is more accurate. Most, but not all characters in an East Asian character set do require more than one byte.

For details about how SAS uses DBCS encodings, see Chapter 21, “SAS System Options for Processing DBCS Data,” on page 711.

MBCS (Multi-Byte Character Set)
is used as a synonym for DBCS.
Common Encoding Methods

An encoding results from applying an encoding method to a coded character set.

An *encoding method* is the application of established industry rules to a coded character set to produce an encoded character scheme. Such rules prescribe the number of bits required for storing the numeric representation of a specific character and its code position in the encoding. ISO 2022 and UTF-8 are examples of encoding methods. For more information, see “Standards Organizations for NLS Encodings” on page 15.

An encoding method is a set of rules that assign numeric representations to a set of characters. These rules govern the size of the encoding, the number of bits used to store the numeric representation of the characters, and the ranges in the encoding where characters appear.

Common encoding methods are listed here:

**ASCII (American Standard Code for Information Interchange)**

is a 7-bit encoding for the United States that provides 128 character combinations. The encoding contains characters for uppercase and lowercase English, American English punctuation, base 10 numbers, and a few control characters. This set of 128 characters is common to most other encodings. ASCII is used by personal computers.

**EBCDIC (Extended Binary Coded Decimal Interchange Code) family**

is an 8-bit encoding that provides 256 character combinations. There are multiple EBCDIC-based encodings. EBCDIC is used on IBM mainframes and most IBM mid-range computers. EBCDIC follows ISO 646 conventions to facilitate translations between EBCDIC encodings and 7-bit (and 8-bit) ASCII-based encodings. The 95 EBCDIC graphical characters include 82 invariant characters (including a blank space), which occupy the same code positions across most EBCDIC single-byte code pages. It also includes 13 variant graphic characters, which occupy varying code positions across most EBCDIC single-byte code pages. For more information about variant characters, see “Code Point Discrepancies among EBCDIC Encodings” on page 16.

There are also multi-byte character set (MBCS) EBCDIC encodings.

**ISO (International Organization for Standardization) 646 family**

is a 7-bit encoding that is an international standard and provides 128 character combinations. The ISO 646 family of encodings is similar to ASCII except that it has 12 code points for national variants. The 12 national variants represent specific characters that are needed for a particular language.

**ISO 8859 family and Windows family**

is an 8-bit extension of ASCII that supports all of the ASCII code points and adds 12 more, providing 256 character combinations. Latin1, which is officially named ISO-8859-1, is the most frequently used member of the ISO 8859 family of encodings. In addition to the ASCII characters, Latin1 contains accented characters, other letters needed for languages of Western Europe, and some special characters. HTTP and HTML protocols are based on Unicode.

**Other encodings**

The ISO 8859 family has other members that are designed for other languages. The following table describes the other encodings that are approved by ISO.
<table>
<thead>
<tr>
<th>ISO Standard</th>
<th>Name of Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 8859-1</td>
<td>Latin 1</td>
<td>US and Western European</td>
</tr>
<tr>
<td>ISO 8859-2</td>
<td>Latin 2</td>
<td>Central and Eastern European</td>
</tr>
<tr>
<td>ISO 8859-3</td>
<td>Latin 3</td>
<td>Southern European, Maltese, and Esperanto</td>
</tr>
<tr>
<td>ISO 8859-4</td>
<td>Baltic</td>
<td>Northern European</td>
</tr>
<tr>
<td>ISO 8859-5</td>
<td>Cyrillic</td>
<td>Slavic languages</td>
</tr>
<tr>
<td>ISO 8859-6</td>
<td>Arabic</td>
<td>Arabic</td>
</tr>
<tr>
<td>ISO 8859-7</td>
<td>Greek</td>
<td>Modern Greek</td>
</tr>
<tr>
<td>ISO 8859-8</td>
<td>Hebrew</td>
<td>Hebrew and Yiddish</td>
</tr>
<tr>
<td>ISO 8859-9</td>
<td>Turkish</td>
<td>Turkish</td>
</tr>
<tr>
<td>ISO 8859-10</td>
<td>Latin 6</td>
<td>Nordic (Inuit, Sámi, Icelandic)</td>
</tr>
<tr>
<td>ISO 8859-11</td>
<td>Latin/Thai</td>
<td>Thai</td>
</tr>
<tr>
<td>ISO 8859-13</td>
<td>Latin 7</td>
<td>Baltic Rim</td>
</tr>
<tr>
<td>ISO 8859-14</td>
<td>Latin 8</td>
<td>Celtic</td>
</tr>
<tr>
<td>ISO 8859-15</td>
<td>Latin 9</td>
<td>Western European</td>
</tr>
<tr>
<td>ISO-8859-16</td>
<td>Latin 10</td>
<td>Southeast European</td>
</tr>
</tbody>
</table>

Unicode is a common and popular character set.

Unicode provides up to 110,116 character combinations. Unicode can accommodate basically all of the world's languages. Unicode is a coded character set. The following are encodings of the Unicode character set:

UTF-8 is an MBCS encoding that contains the Latin-script languages, Greek, Cyrillic, Arabic, and Hebrew. It also includes East Asian languages such as Japanese, Chinese, and Korean. The characters in UTF-8 are of varying width, from 1 to 4 bytes. UTF-8 maintains ASCII compatibility by preserving the ASCII characters in code positions 1 through 128.

UTF-16 is a 16-bit form that contains all of the most common characters in all modern writing systems. Most of the characters are uniformly represented with 2 bytes,
although there is extended space, called surrogate space, for additional characters that require 4 bytes.

UTF-32
is a 32-bit form whose characters each occupy 4 bytes.

Also, a number of encoding standards have been developed for East Asian languages. Some of these are listed in the following table.

<table>
<thead>
<tr>
<th>Standard</th>
<th>Name of Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB 2312-80</td>
<td>Simplified Chinese</td>
<td>People's Republic of China</td>
</tr>
<tr>
<td>CNS 11643</td>
<td>Traditional Chinese</td>
<td>Taiwan</td>
</tr>
<tr>
<td>Big-5</td>
<td>Traditional Chinese</td>
<td>Taiwan</td>
</tr>
<tr>
<td>KS C 5601</td>
<td>Korean National Standard</td>
<td>Korea</td>
</tr>
<tr>
<td>JIS</td>
<td>Japan Industry Standard</td>
<td>Japan</td>
</tr>
<tr>
<td>Shift-JIS</td>
<td>Japan Industry Standard multibyte encoding</td>
<td>Japan</td>
</tr>
</tbody>
</table>

There are other encodings in the standards for EBCDIC and Windows that support different languages and locales.

Standards Organizations for NLS Encodings

Encodings that are supported by SAS are defined by the following standards organizations:

International Organization for Standardization (ISO)
promotes the development of standardization and related activities to facilitate the free flow of goods and services between nations and to advocate for the exchange of intellectual, scientific, and technological information. ISO also establishes standards for encodings.

American National Standards Institute (ANSI)
coordinates voluntary standards and conformity to those standards in the United States. ANSI works with ISO to establish global standards.

Unicode Consortium
that develops and promotes the Unicode standard, which provides a unique number for every character.
Code Point Discrepancies among EBCDIC Encodings

Selected characters do not occupy the same code point locations in code maps for all EBCDIC encoding methods. For example, the following characters occupy different code point locations in the respective EBCDIC code maps for U.S. English and German.

Table 3.3  EBCDIC Code Point Discrepancies for Selected Languages

<table>
<thead>
<tr>
<th>EBCDIC Code Points</th>
<th>U.S. English</th>
<th>Finnish</th>
<th>Spanish</th>
<th>Austrian/German</th>
</tr>
</thead>
<tbody>
<tr>
<td>4A</td>
<td>€</td>
<td>§</td>
<td>[</td>
<td>Ä</td>
</tr>
<tr>
<td>4F</td>
<td></td>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td>5A</td>
<td>!</td>
<td>□</td>
<td>]</td>
<td>Ü</td>
</tr>
<tr>
<td>5B</td>
<td>$</td>
<td>Å</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>5F</td>
<td>~</td>
<td>^</td>
<td>~</td>
<td>^</td>
</tr>
<tr>
<td>6A</td>
<td></td>
<td></td>
<td>ö</td>
<td>ñ</td>
</tr>
<tr>
<td>79</td>
<td>’</td>
<td>é</td>
<td>’</td>
<td>’</td>
</tr>
<tr>
<td>7B</td>
<td>#</td>
<td>Ä</td>
<td>Ń</td>
<td>#</td>
</tr>
<tr>
<td>7C</td>
<td>@</td>
<td>Ö</td>
<td>@</td>
<td>§</td>
</tr>
<tr>
<td>A1</td>
<td>~</td>
<td>ü</td>
<td>~</td>
<td>ß</td>
</tr>
<tr>
<td>C0</td>
<td>{</td>
<td>ä</td>
<td>{</td>
<td>ä</td>
</tr>
<tr>
<td>D0</td>
<td>}</td>
<td>å</td>
<td>}</td>
<td>ü</td>
</tr>
<tr>
<td>E0</td>
<td>\</td>
<td>É</td>
<td>\</td>
<td>Ö</td>
</tr>
</tbody>
</table>

Examples of characters that are commonly used in programming languages are { and $. These characters are known as variant characters. For example, if a German mainframe user entered an ä, which occupies code point C0, an American compiler would interpret code point C0 as a {.
Collating Sequence

Overview to Collating Sequence

The collating sequence is the order in which characters are sorted. For example, when the SORT procedure is executed, the collating sequence determines the sort order (higher, lower, or equal to) of a particular character in relation to other characters.

The default collating sequence is binary collation, which sorts characters according to each character's location in the code page of the session encoding. (The session encoding is the default encoding for a SAS session. The default encoding can be specified by using various SAS language elements.) The sort order corresponds directly to the arrangement of the code points within the code page. The two single-byte character encodings that data processing uses most widely are ASCII and EBCDIC. The OpenVMS, UNIX, and Windows operating environments use ASCII encodings; IBM mainframe computers use EBCDIC encodings.

Binary collation is the fastest type of collation because it is the most efficient for the computer. However, locating characters within a binary-collated report might be difficult if you are not familiar with this method. For example, a binary-collated report lists words beginning with uppercase characters separately from words beginning with lowercase characters, and words beginning with accented characters after words beginning with unaccented characters. Therefore, for ASCII-based encodings, the capital letter Z precedes the lowercase letter a. Similarly, for EBCDIC-based encodings, the lowercase letter z precedes the capital letter A.

You can request an alternate collating sequence that overrides the binary collation. To request an alternate collating sequence, specify one of the following sequences:

- a translation table name
- an encoding value
- linguistic collation

Table 3.4 on page 17 illustrates the results of using different collating sequences to sort a short list of words:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Translation Table</th>
<th>Linguistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aaron</td>
<td>aardvark</td>
<td>aardvark</td>
</tr>
<tr>
<td>Aztec</td>
<td>azimuth</td>
<td>Aaron</td>
</tr>
<tr>
<td>Zeus</td>
<td>Aaron</td>
<td>azimuth</td>
</tr>
<tr>
<td>aardvark</td>
<td>Aztec</td>
<td>Aztec</td>
</tr>
<tr>
<td>azimuth</td>
<td>cote</td>
<td>cote</td>
</tr>
<tr>
<td>cote</td>
<td>coté</td>
<td>côte</td>
</tr>
</tbody>
</table>
The first column shows the results of binary collation on characters that are represented
in an ASCII-based encoding. The alphabetization is not consistent because of the
separate grouping of words that begin with uppercase and lowercase characters. For
example, the word Zeus appears before aardvark because of the code points that are
assigned to the characters within the ASCII-based encoding.

The second column shows the results of specifying a translation table that alternates the
ordering of lowercase and uppercase characters. If you use the translation table, the word
aardvark appears before Zeus. However, the word azimuth appears before Aaron because
the translation table assigns a weight value to the lowercase character \textit{a} that is less than
the weight value of the uppercase character \textit{A}. In addition, accents are sorted from left to
right. For example, coté comes before côte.

The third column shows the results of specifying the ASCII-based, double-byte latin1
encoding.

The last column shows the results of linguistic collation for the session locale \textit{fr\_FR}
(French\_France), which uses a collation algorithm to alphabetize words. The algorithm
specifies that words beginning with lowercase characters appear before words beginning
with uppercase characters. In addition, this linguistic collation sorts accents from right to
left because of the French locale specification.

SAS has adopted the International Components for Unicode (ICU) to implement
linguistic collation. The ICU and its implementation of the Unicode Collation Algorithm
(UCA) have become a standard. The collating sequence is the default provided by the
ICU for the specified locale.

\textbf{Request Alternate Collating Sequence}

To request an alternate collating sequence, use the following SAS language elements:

- \texttt{SORTSEQ=} option in the \texttt{PROC SORT} statement. See “Collating Sequence Option”
on page 617.
- \texttt{SORTSEQ=} system option. See “\texttt{SORTSEQ=} System Option: UNIX, Windows, and
z/OS” on page 602.

Note that neither method supports all of the collating sequences. For example, only the
\texttt{SORTSEQ=} option in the \texttt{PROC SORT} statement supports linguistic collation. However,
both the \texttt{SORTSEQ=} option in the \texttt{PROC SORT} statement and the \texttt{SORTSEQ=} system
option support translation table collating sequences.

The \texttt{BASE (V9)} engine and the \texttt{REMOTE} engine for \texttt{SAS/SHARE} support all alternate
collating sequences. The \texttt{V9TAPE} sequential engine supports the use of a translation
Specifying a Translation Table

A translation table is a SAS catalog entry that transcodes data from one single-byte encoding to another single-byte encoding. A translation table also reorders characters when sorting them. A translation table can be one that SAS provides, such as a standard collating sequence like ASCII, EBCDIC, or DANISH; or it can be a user-defined translation table.

When you specify a translation table for an alternate collating sequence, the characters are reordered by mapping the code point of each character to an integer weight value in the range of 0 to 255. A binary collation is then performed.

For collating purposes, you can create translation tables that order characters so that lowercase and uppercase characters alternate. For example, you can create a translation table to correct the situation in which z precedes a in an ASCII-based encoding. (However, regardless of the weight assignments in the translation table, it is difficult to achieve a true alphabetic ordering that takes the character case into account.) You can also create a translation table that orders alphabetic characters of a particular language in their expected order.

The TRANTAB procedure creates, edits, and displays translation tables. For example, you can display a translation table to view the character-weight values. The translation tables that are supplied by SAS are stored in the SASHELP.HOST catalog. Any translation table that you create or customize is stored in your SASUSER.PROFILE catalog. Translation tables have an entry type of TRANTAB. See Chapter 19, “TRANTAB Procedure,” on page 669 for more information about translation tables.

You can specify a translation table with the SORTSEQ= option in the PROC SORT statement or with the SORTSEQ= system option. For example, if your operating environment sorts with the ASCII-based WLatin1 encoding by default, and you want to sort with a translation table that alternates uppercase and lowercase characters, issue the following statements to specify the SAS translation table FRSOLAT1:

```sas
proc sort data=myfiles.test sortseq=FRSOLAT1;
   by name;
run;
```

A SAS data set that is sorted with a translation table contains a sort indicator that displays the specified translation table name as the collating sequence in CONTENTS procedure output.

Specifying an Encoding Value

An encoding is a set of characters (letters, logograms, digits, punctuation marks, symbols, and control characters) that have been mapped to hexadecimal values, called code points, that computers use. When you specify an encoding value for an alternate collating sequence, the characters are transcoded from the SAS session encoding to the specified encoding, and then a binary collation is performed. You can specify all encoding values that are supported by the ENCODING= option, including multi-byte encodings. Note that specifying a translation table can transcode data, but translation tables are limited to single-byte encodings.

You can specify an encoding value with the SORTSEQ= option in the PROC SORT statement, but you cannot specify an encoding value in the SORTSEQ= system option. For example, you want to sort a SAS data set and then transport it to a Japanese
Windows environment. If your session encoding is ASCII-based and binary collation is in effect, you can issue the following statements to specify the ASCII-based double-byte encoding SHIFT-JIS:

```sas
proc sort data=myfiles.test sortseq='shift-jis';
  by name;
run;
```

Note that SAS checks the encoding value for any translation tables with the same name. If a translation table name exists, SAS uses the translation table.

A SAS data set that is sorted with an encoding value contains a sort indicator that displays the specified encoding value as the collating sequence in CONTENTS procedure output.

**Specifying Linguistic Collation**

Linguistic collation sorts characters according to rules of language and produces results that are intuitive and culturally acceptable. The results are similar to the collation used in printed materials such as dictionaries, phone books, and book indexes. Linguistic collation is useful for generating reports or other data presentations and for achieving compatibility between systems.

SAS incorporates the International Components for Unicode (ICU), which is an open-source library that provides routines for linguistic collation that are compatible with the Unicode Collation Algorithm (UCA). The UCA is a standard by which Unicode strings can be compared and ordered.

To request linguistic collation, you must use the SORTSEQ= option in the PROC SORT statement because the SORTSEQ= system option does not support linguistic collation. For example, the following statements cause the SORT procedure to collate linguistically, in accordance with the French_France locale:

```sas
options locale=fr_FR;
proc sort data=myfiles.test sortseq=linguistic;
  by name;
run;
```

When linguistic collation is requested, SAS uses the default linguistic collation algorithm that is provided by the ICU for the SAS session locale. This algorithm reflects the language, local conventions such as data formatting, and culture for a geographical region. You can modify the algorithm by specifying options in parentheses following the LINGUISTIC keyword. For example, you can specify a different locale; you can specify the CASE_FIRST= option to collate lowercase characters before uppercase characters, or vice versa; and so on. Generally, it is not necessary to specify options, because the ICU associates defaults with the various languages and locales. For more information about the linguistic options, see the SORTSEQ= option in “Collating Sequence Option” on page 617 or the SORTSEQ= option in the PROC SORT statement in *Base SAS Procedures Guide*.

A SAS data set that is sorted linguistically contains a sort indicator that displays the collating sequence LINGUISTIC in CONTENTS procedure output. Along with the sort indicator, the data set also records a complete description of the linguistic collating sequence in the file's descriptor information, which is also displayed in CONTENTS procedure output.
Determining the Encoding of a SAS Session and a Data Set

Encoding of a SAS Session

To determine your current SAS session encoding, which is the value assigned to the ENCODING= system option, you can use the OPTIONS procedure or the OPTIONS window. For example, the following PROC OPTIONS statement displays the session encoding value:

```sas
proc options option=encoding;
run;
```

The SAS log displays the following information:

ENCODING=WLATIN1 Specifies default encoding for processing external data.

You can also determine the SAS Session Encoding by using the following command:

```sas
%PUT %SYSFUNC(getOption(ENCODING));
```

Encoding of a SAS Data Set

To determine the encoding of a specific SAS data set, follow these steps:

1. Locate the data set using SAS Explorer.
2. Right-click the data set.
3. Select Properties from the menu.
4. Click the Details tab.

The encoding of the data set is listed, along with other information.

You can also determine the encoding by using the following command:

```sas
%LET DSID=%SYSFUNC(open(sashelp.class,i));
%PUT %SYSFUNC(ATTRC(&DSID,ENCODING));
```

You can display the encoding of any SAS 9 data set by using the CONTENTS procedure or the Properties window in the SAS windowing environment.

An example follows of output that is reported from the CONTENT procedure in the SAS log. The encoding is Western latin1.
**Output 3.1  Encoding Reported in the SAS Log**

The SAS System 10:15 Friday, June 06, 2003 1

The CONTENTS Procedure

Data Set Name WORK.GRADES  Observations 1

Member Type DATA  Variables 4

Engine V9  Indexes 0

Created 11:03 Friday, June 06 2003  Observation Length 32

Last Modified 11:03 Friday, June 06, 2003  Deleted Observations 0

Protection

Data Set Type

NO  Sorted

Label

Data Representation HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64

Encoding latin1

Western (ISO)

Engine/Host Dependent Information

Data Set Page Size 4096

Number of Data Set Pages 1

First Data Page 1

Max Obs per Page 126

Obs in First Data Page 1

Number of Data Set Repairs 0

File Name C:\TEMP\SAS Temporary

Files TD228\grades.sas7bdat

Release Created 9.0000M0

Host Created WIN_NT

Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>final</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>student</td>
<td>Char</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>test1</td>
<td>Num</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>test2</td>
<td>Num</td>
<td>8</td>
</tr>
</tbody>
</table>

**Default SAS Session Encoding**

The ENCODING= option is used to specify the SAS session encoding, which establishes the environment to process SAS syntax and SAS data sets, and to read and write external files. If neither the LOCALE= nor ENCODING= options is set, a default value is set.
Table 3.5 Default SAS Session Encoding Values

<table>
<thead>
<tr>
<th>Operating Environment</th>
<th>Default ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS</td>
<td>OPEN ED-1047</td>
<td>OpenEdition EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>UNIX</td>
<td>Latin1</td>
<td>Western (ISO)</td>
</tr>
<tr>
<td>Windows</td>
<td>WLatin1</td>
<td>Western (Windows)</td>
</tr>
</tbody>
</table>

For a complete list of supported encoding values for a SAS session, see Chapter 23, “Encoding Values for a SAS Session,” on page 723.

Setting the Encoding of a SAS Session

You can set the session encoding by using the ENCODING= system option, the DBCS options, or the LOCALE= system option.

Note: Values for the ENCODING= system option depend on the operating environment.

The priority order for setting the encoding is as follows:

1. ENCODING= system option
   
The SAS session encoding is determined by the ENCODING= option regardless of whether the DBCS or LOCALE= options are specified. If the ENCODING= option is specified, a set of valid DBCS options is set regardless of whether the user has specified those options. Also, if the ENCODING= option is specified, the LOCALE= option is set to an appropriate value unless a value has been specified by the user.

   Note: If the ENCODING= option is specified, the TRANTAB= option is implicitly set. TRANTAB options apply only to external files.

2. DBCS options
   
   Most North and South American, European, Middle Eastern, African, and Australian users use the SAS SBCS environment and do not use the DBCS environment.

   If the ENCODING= option is not specified, the SAS session encoding is determined by the DBCS options regardless of whether the LOCALE= option is specified. The LOCALE= option is set to an appropriate value unless a value has been specified by the user.

   The encoding is determined by the values of the DBCSLANG and DBCSTYPE options for DBCS languages, such as Japanese, Korean, Simplified Chinese, and Traditional Chinese.

   The DBCS options are valid only when the DBCS extension directory is included in the path option list. The path of the DBCS extension dynamic link library (DLLs) has to be located at the top of the pathname list of the path option for the DBCS languages when you want to invoke a DBCS SAS session. The DBCS extension DLLs are located in the directory `!SASROOT/dbcs/sasexe` by default.

   Also, you might have to specify the resourcesloc, msg, and sashelp options to use localized resources even if the SAS session encoding is not a DBCS language (for
example, Polish, German, and French). The localized resources are located under `SASROOT/nls/<language identifier>/<sasmsg, sashelp, sasmacro, resource>`. The values for language identifiers are: cs, de, en, es, fr, hu, it, ja, ko, pl, ru, sv, zh, and zt.

You can specify a sasv9.cfg file located in the localized directories such as `SASROOT/nls/<language identifier>` so that you do not have to consider using the `path`, `resourcesloc`, `sasmsg`, and `sashelp` options.

If DBCS (which specifies that SAS process DBCS encodings) is specified, `DBCSLANG= and DBCSTYPE=` options are implicitly set. The default values for `DBCSTYPE=` and `DBCSLANG=` match those values for the DBCS environment on the host (for example, Japanese, Korean, or Chinese).

3. LOCALE= system option

The SAS session encoding is determined by the `LOCALE=` option and the platform, if the `ENCODING=` or DBCS options are not specified.

The following example shows that encoding is explicitly set by default for the Spanish_Spain locale:

```
sas9 -locale Spanish_Spain
```

The wlatin1 encoding is the default encoding for the Spanish_Spain locale.

The following example shows that the wlatin2 encoding is set explicitly when SAS is invoked:

```
sas9 -encoding wlatin2
```

*Note:* Setting DBCS encodings, DBCS options, or a CJK (Chinese, Japanese, Korean) locale on SAS if the DBCS extensions are not available fails to successfully invoke SAS.

*Note:* Changing the encoding for a SAS session does not affect SAS keywords or SAS log output, which remain in English.

In Table 3.6 on page 24, the following values for the CJK locales are based on locale and platform:

**Table 3.6** Default Encoding Values Based on the LOCALE= Option

<table>
<thead>
<tr>
<th>Locales</th>
<th>WIN</th>
<th>MVS</th>
<th>UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>zh_TW</td>
<td>MS-950 (ywin)</td>
<td>IBM-937 (yibm)</td>
<td>Solaris on X64, Solaris on SPARC, EUC-TW (yeuc) others: MS-950 (ywin)</td>
</tr>
<tr>
<td>zh_HK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zh_MO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zh_CN</td>
<td>EUC-CN (zeuc)</td>
<td>IBM-935 (zibm)</td>
<td>EUC-CN (zeuc)</td>
</tr>
<tr>
<td>zh_SG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ja_JP</td>
<td>SHIFT-JIS (sjis)</td>
<td>IBM-939 (jibm)</td>
<td>h64, h6i, AIX on Power, SHIFT-JIS (sjis) others: EUC-JP (jieuc)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM-930(j930)</td>
<td></td>
</tr>
</tbody>
</table>
Encoding Behavior in a SAS Session

Encoding Support for Data Sets by SAS Release

For Base SAS files, there are three categories of encoding support, which is based on the version of SAS that created the file:

- Data sets that are created in SAS 9 automatically have an encoding attribute, which is specified in the descriptor portion of the file. In SAS 9, DBCS recognizes the DBCSTYPE value and converts it to the encoding value and specifies it in the descriptor portion of the field, by default.

- Data sets that are created in SAS 7 and SAS 8 do not have an encoding value that is specified in the file. It is assumed that SAS 7 and SAS 8 data sets were created in the SAS session encoding of the operating environment. However, the descriptor portion of the file does support an encoding value. When you replace or update a SAS 7 or SAS 8 file in a SAS 9 session, SAS specifies the current session encoding in the descriptor portion of the file, by default. In SAS 8, DBCS has the DBCSTYPE field, instead of the encoding field.

- Data sets created in SAS 6 do not have an encoding value that is associated with the file. An encoding value cannot be specified in the file.

Output Processing

When you create a data set in SAS 9, encoding is determined as follows:

- If a new output file is created, the data is written to the file using the current session encoding.

- If a new output file is created using the OUTREP= option, which specifies a data representation that is different from the current session, the data is written to the file using the default session encoding for the operating system that is specified by the OUTREP= value. For more information, see “OUTREP= Data Set Option” on page 60.

- If a new output file replaces an existing file, the new file inherits the encoding of the existing file. For output processing that replaces an existing file that is from another operating environment or if the existing file has no encoding that is specified in it, then the current session encoding is used.

Input Processing

For input (read) processing in SAS 9, encoding behavior is as follows:

- Most users choose the default behavior that does not specify an encoding for the input file.
• If the session encoding and the encoding that is specified in the file are incompatible, the data is transcoded to the session encoding. For example, if the current session encoding is ASCII and the encoding that is specified in the file is EBCDIC, SAS transcodes the data from EBCDIC to ASCII.

• If a file does not have an encoding specified in it, SAS transcodes the data only if the file's data representation is different from the current session.

**Reading and Writing External Files**

SAS reads and writes external files using the current session encoding. SAS assumes that the external file has the same encoding as the session encoding. For example, if you are creating a new SAS data set by reading an external file, SAS assumes that the encoding of the external file and the current session are the same. If the encodings are not the same, the external data could be written incorrectly to the new SAS data set. You need to specify an appropriate ENCODING option. The following example specifies the Shift-JIS encoding:

```sas
filename in 'external-file'
  encoding='Shift-JIS';
data mylib.contacts;
infile in;
length name $ 30 first $ 30 street $ 60 zip $ 10 city $ 30;
input name first street zip city;
run;
```

For details about the syntax for the SAS statements that perform input and output processing, see “SAS Options That Transcode SAS Data” on page 30.
Overview to Transcoding

Transcoding is the process of converting data from one encoding to another. Transcoding is necessary when the SAS session encoding and the encoding of the data are different. Transcoding is often necessary when you move data between operating environments that use different locales and encoding.

For example, consider a file that was created under a UNIX operating environment that uses the Latin1 encoding, then moved to an IBM mainframe that uses the German EBCDIC encoding. When the file is processed on the IBM mainframe, the data is remapped from the Latin1 encoding to the German EBCDIC encoding. If the data contains an uppercase letter Ä, the hexadecimal number is converted from C4 to 4A.

Transcoding does not translate between languages; transcoding remaps characters.

In order to dynamically transcode data between operating environments that use different encodings, an explicit encoding value must be specified. For details, see Chapter 22, “Encoding Values in SAS Language Elements,” on page 713.
Common Reasons for Transcoding

Some situations where data might commonly be transcoded are:

- when you share data between two different SAS sessions that are running in different locales or in different operating environments
- when you copy and paste data between SAS sessions running in different locales

Transcoding and Translation Tables

Translation tables are implicitly set by the LOCALE= system option. They are used only for transcoding external files. There is direct transcoding for SAS files between the session encodings.

Specifying LOCALE= or ENCODING= indirectly sets the appropriate translation table values in the TRANTAB= option. Translation tables are used for transcoding one SBCS encoding to another and back again. For example, there is a specific translation table that maps Windows Latin2 to ISO Latin2. TRANTABS apply only to external files.

The following figure shows a translation table. The area of a translation table for mapping from Windows Latin 2 (wlt2) to ISO Latin 2 (lat2) is named "table 1," and the area for mapping characters from ISO Latin 2 to Windows Latin 2 is named "table 2."
CAUTION:

Do not change a translation table unless you are familiar with its purpose.

Translation tables are used internally by the SAS supervisor to implement NLS. If you are unfamiliar with the purpose of translation tables, do not change the specifications without proper technical advice.

The TRANTAB= option specifies the translation table to be used in the SAS session. For details, see “TRANTAB= System Option” on page 605. The TRANTAB procedure is used to create, edit, and display customized translation tables. For details, see Chapter 19, “TRANTAB Procedure,” on page 669.
SAS Options That Transcode SAS Data

The following SAS options for various language elements enable you to transcode, or to override the default encoding behavior. These elements enable you to specify a different encoding for a SAS file or a SAS application or to suppress transcoding.

Table 4.1  SAS Options That Transcode SAS Data

<table>
<thead>
<tr>
<th>Option</th>
<th>Where Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARSET=</td>
<td>ODS MARKUP statement</td>
</tr>
<tr>
<td>CORRECTENCODING=</td>
<td>MODIFY statement of the DATASETS procedure</td>
</tr>
<tr>
<td>ENCODING=</td>
<td>%INCLUDE, FILE, FILENAME, INFILE, ODS statements; FILE and INCLUDE commands</td>
</tr>
<tr>
<td>ENCODING=</td>
<td>in a DATA step</td>
</tr>
<tr>
<td>INENCODING=</td>
<td>LIBNAME statement</td>
</tr>
<tr>
<td>ODSCHARSET=</td>
<td>LIBNAME statement for XML</td>
</tr>
<tr>
<td>ODSTRANTAB=</td>
<td>LIBNAME statement for XML</td>
</tr>
<tr>
<td>OUTENCODING=</td>
<td>LIBNAME statement</td>
</tr>
<tr>
<td>XMLENCODING=</td>
<td>LIBNAME statement for XML</td>
</tr>
</tbody>
</table>

For a list of supported encoding values to use for these options, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

Transcoding between Operating Environments

Transcoding occurs automatically when SAS files are moved or accessed across operating environments. Common SAS transcoding activities include:

CPORT and CIMPORT procedures
To create a transport file, the data is first converted from the source encoding to transport format, then the data is converted from the transport format to the target encoding. For details, see *Base SAS Procedures Guide*.

CEDA (cross environment data access) feature of SAS
when you process a SAS data set that has an encoding that is different from the current session encoding, SAS automatically uses CEDA software to transcode data. (CEDA also converts a SAS file to the correct data representation when you move a
file between operating environments.) For details, see *SAS Language Reference: Concepts*.

SAS/CONNECT Data Transfer Services (UPLOAD and DOWNLOAD procedures)
For details, see *SAS/CONNECT User’s Guide*.

SAS/CONNECT Compute Services (RSUBMIT statement)
identifies a block of statements that a client session submits to server session for processing. For details, see *SAS/CONNECT User’s Guide*.

SAS/CONNECT and SAS/SHARE Remote Library Services (LIBNAME)
References a library on a remote machine for client access. For details, see *SAS/CONNECT User’s Guide* and *SAS/SHARE User’s Guide*.

---

**Transcoding Considerations**

Although transcoding usually occurs with no problems, there are situations that can affect your data and produce unsatisfactory results. For example:

- Encodings can conflict with another. That is, two encodings can use different code points for the same character, or use the same code points for two different characters.

- Characters in one encoding might not be present in another encoding. For example, a specific encoding might not have a character for the dollar sign ($). Transcoding the data to an encoding that does not support the dollar sign would result in the character not printing or displaying.

- The number of bytes for a character in one encoding can be different from the number of bytes for the same character in another encoding. An example is transcoding from a DBCS to an SBCS. Therefore, transcoding can result in character value truncation.

- If an error occurs during transcoding such that the data cannot be transcoded back to its original encoding, data can be lost. That is, if you open a data set for update processing, the observation might not be updated. However, if you open the data set for input (read) processing and no output data set is open, SAS issues a warning that can be printed. Processing proceeds and allows a PRINT procedure or other Read operation to show the data that does not transcode.

- CEDA has some processing limitations. For example, CEDA does not support update processing.

- Incorrect encoding can be stamped on a SAS 7 or SAS 8 data set if it is copied or replaced in a SAS 9 session with a different session encoding from the data. The incorrect encoding stamp can be corrected with the CORRECTENCODING= option in the MODIFY statement in PROC DATASETS. If a character variable contains binary data, transcoding might corrupt the data.
Compatible and Incompatible Encodings

Overview to Compatible and Incompatible Encodings

ASCII is the foundation for most encodings, and is used by most personal computers, minicomputers, and workstations. However, the IBM mainframe uses an EBCDIC encoding. Therefore, ASCII and EBCDIC machines and data are incompatible. Transcoding is necessary if some or all characters in one encoding are different from the characters in the other encoding.

However, to avoid transcoding, you can create a data set and specify an encoding value that SAS does not transcode. For example, if you use the following values in either the ENCODING= data set option, or the INENCODING=, or the OUTENCODING= option in the LIBNAME statement, transcoding is not performed:

- ANY specifies that no transcoding is desired, even between EBCDIC and ASCII encodings.
  
  Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

- ASCIIANY enables you to create a data set that is compatible with all ASCII-based encodings.

- EBCDICANY enables you to create a data set that is compatible with all EBCDIC-based encodings.

You might want to create a SAS data set that contains mixed encodings. An example is both Latin1 and Latin2. You do not want the data transcoded for either input or output processing. By default, data is transcoded to the current session encoding.

Data must be transcoded when the SAS file and the SAS session use incompatible encodings. An example is ASCII and EBCDIC.

In some cases, transcoding is not required because the SAS file and the SAS session have compatible encodings.

For a list of the encodings, by operating environment, see Chapter 23, “Encoding Values for a SAS Session,” on page 723.

Line-feed Characters and Transferring Data between EBCDIC and ASCII

Software that runs under ASCII operating environments requires the end of the line be specified by the line-feed character. When data is transferred from z/OS to a machine that supports ASCII encodings, formatting problems can occur, particularly in HTML output, because the EBCDIC newline character is not recognized. SAS supports two sets of EBCDIC-based encodings for z/OS:

- The encodings that have EBCDIC in their names use the traditional mapping of EBCDIC line-feed to ASCII line-feed character, which can cause data to appear as one stream.

- The encodings that have Open Edition in their names use the line-feed character as the end-of-line character. When the data is transferred to an operating environment that uses ASCII, the EBCDIC newline character maps to an ASCII line-feed
character. This mapping enables ASCII applications to interpret the end-of-line correctly, resulting in better formatting.

For a list of the encodings, by operating environment, see Chapter 23, “Encoding Values for a SAS Session,” on page 723.

**EBCDIC and OpenEdition Encodings Are Compatible**

EBCDIC and OpenEdition are compatible encodings.

Encodings that contain EBCDIC in their names use the traditional mapping of EBCDIC line-feed (0x25) and newline (0x15) characters.

Encodings that contain OPEN_ED in their names and OpenEdition in their descriptions switch the mapping of the newline and line-feed characters. That is, they use the line-feed character as the end-of-line character.

If the two encodings use the same code page number but one is EBCDIC and the other is Open Edition, no transcoding is necessary.

Example:

If the data is encoded in EBCDIC1143 and the SAS session is encoded in OPEN_ED-1143, no transcoding is necessary because they use the same 1143 code page.

In order to transfer data between ASCII and EBCDIC, you can specify Open Edition encodings from the list of compatible encodings.

**Some East Asian MBCS Encodings Are Compatible**

Some East Asian double-byte (DBCS) are compatible encodings. Each line in the list contains compatible encodings:

- SHIFT-JIS, MS-932, IBM-942, MACOS-1
- MS-949, MACOS-3, EUC-KR
- EUC-CN, MS-936, MACOS-25, DEC-CN
- EUC-TW, DEC-TW
- MS-950, MACOS-2, BIG5

If the SAS session is encoded in one of the encodings in the group and the data set is encoded in another encoding, but in the same group, then no transcoding occurs.

Example:

If the session encoding is SHIFT-JIS and the data set encoding is IBM-942, then no transcoding occurs.

---

**Preventing Transcoding**

Some encoding values enable you to create a data set that SAS does not transcode. You might not want to transcode data for input or output processing but rather you might want to create a SAS library that contains data in mixed encodings. An example is both Latin1 and Latin2.
For example, you can avoid transcoding if you use the following values in either the ENCODING= data set option or the INENCODING= or OUTENCODING= options in the LIBNAME statement:

- **ANY** specifies that no transcoding is desired, even between EBCDIC and ASCII encodings.

  Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

- **ASCIIANY** specifies that no transcoding is required between any ASCII-based encodings.

- **EBCDICANY** specifies that no transcoding is required between any EBCDIC-based encodings.

For details, see “ENCODING= Data Set Option” on page 57 and “INENCODING= and OUTENCODING= Options” on page 634.

You can prevent transcoding for a specific column of data while the rest of the character data in the data set is transcoded by using the TRANSCODE= option. For more information, see “TRANSCODE= Column Modifier on PROC SQL” on page 637.

### Avoiding Character Data Truncation By Using the CVP Engine

When you specify the ENCODING= data set option, the encoding for the output data set might require more space than the original data set. For example, when writing DBCS data in a Windows environment using the UTF8 encoding, each DBCS character might require three bytes. To avoid data truncation, each variable must have a width that is 1.5 times greater than the width of the original data.

When you process a SAS data file that requires transcoding, you can request that the CVP (character variable padding) engine expand character variable lengths so that character data truncation does not occur. (A variable's length is the number of bytes used to store each of the variable's values.)

Character data truncation can occur when the number of bytes for a character in one encoding is different from the number of bytes for the same character in another encoding, such as when a single-byte character set (SBCS) is transcoded to a double-byte character set (DBCS) or to a multi-byte character set (MBCS). An SBCS represents each character in one byte, and a DBCS represents each character in two bytes. An MBCS represents characters in a varying length from one to four bytes. For example, when transcoding from Wlatin2 to a Unicode encoding, such as UTF-8, the variable lengths (in bytes) might not be sufficient to hold the values, and the result is character data truncation.

Using the CVP engine, you specify an expansion amount so that variable lengths are expanded before transcoding, then the data is processed. Think of the CVP engine as an intermediate engine that is used to prepare the data for transcoding. After the lengths are increased, the primary engine, such as the default base engine, is used to do the actual file processing.

The CVP engine is a read-only engine for SAS data files only. You can request character variable expansion (for example, with the LIBNAME statement) in either of the following ways:
• explicitly specify the CVP engine and using the default expansion of 1.5 times the variable lengths.

• implicitly specifying the CVP engine with the LIBNAME options CVPBYTES= or CVPMULTIPLIER=. The options specify the expansion amount. In addition, you can use the CVPENGINE= option to specify the primary engine to use for processing the SAS file; the default is the default SAS engine.

For example, the following LIBNAME statement explicitly assigns the CVP engine. Character variable lengths are increased using the default expansion, which multiples the lengths by 1.5. For example, a character variable with a length of 10 has a new length of 15, and a character variable with a length of 100 has a new length of 150:

```sas
libname expand cvp ' SAS data-library';
```

*Note:* The expansion amount must be large enough to accommodate any expansion. Otherwise, truncation still occurs.

*Note:* For processing that conditionally selects a subset of observations by using a WHERE expression, using the CVP engine might affect performance. Processing the file without using the CVP engine might be faster than processing the file using the CVP engine. For example, if the data set has indexes, the indexes are not used in order to optimize the WHERE expression if you use the CVP engine.

For more information and examples, see “CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options” on page 625.
Overview to Double-Byte Character Sets (DBCS)

Because East Asian languages have thousands of characters, double (two) bytes of information are needed to represent most characters.

Each East Asian language usually has more than one DBCS encoding system, due to nonstandardization among computer manufacturers. SAS processes the DBCS encoding information that is unique to each manufacturer for the major East Asian languages.

With the proper software extensions, you can use SAS for the following functions:

- display any of the major East Asian languages in the DBCS version of the SAS System
- import data from East Asian language computers and move the data from one application or operating environment to another (which might require SAS ACCESS or other SAS products)
- convert standard East Asian date and time notation to SAS date values, SAS time values, and SAS datetime values
- create data sets and various types of output (such as reports and graphs) that contain East Asian language characters.
East Asian Languages

East Asian languages include:

- Chinese, which is written in Simplified Chinese script, and is used in the People's Republic of China and Singapore
- Chinese, which is written in Traditional Chinese script, and is used in Hong Kong Special Administrative Region of the People's Republic of China (SAR), Macau SAR, and Taiwan
- Japanese
- Korean

Specifying DBCS

To specify DBCS, use the following SAS system options:

ENCODING
  recognizes default character-est encoding.

DBCS
  recognizes DBCS characters.

You can set the session encoding by using the ENCODING= system option, the DBCS options, or the LOCALE= system option. The ENCODING= system option has priority. For more information, see “Setting the Encoding of a SAS Session” on page 23. Please see the “ENCODING System Option: UNIX, Windows, and z/OS” on page 587 for more information.

Requirements for Displaying DBCS Character Sets

In order to display data sets that contain DBCS characters, you must have the following resources:

- system support for multiple code pages
- DBCS fonts that correspond to the language that you intend to use

If you need to create a user-defined character for use with SAS software, your computer must support DBCS. These computers have a limited availability in the U.S. and Europe. These East Asian language computer systems use various methods of creating the characters. In one popular method, the user enters the phonetic pronunciation of the character, often using Latin characters. The computer presents a menu of characters whose sounds are similar to the phonetic pronunciation and prompts the user to select one of them.
When You Can Use DBCS Features

After you have set up your SAS session to recognize a specific DBCS language and operating environment, you can work with your specified language in these general areas:

- the DATA step and batch-oriented procedures
- windowing and interactive capabilities
- cross-system connectivity and compatibility
- access to databases
- graphics

In a DATA step and in batch-oriented procedures, you can use DBCS wherever a text string within quotation marks is allowed. Variable values, variable labels, and data set labels can all be in DBCS. DBCS can also be used as input data and with range and label specifications in the FORMAT procedure. In WHERE expression processing, you can search for embedded DBCS text.

DBCS and SAS on a Mainframe

Another type of DBCS encoding exists on mainframe systems, which combine DBCS support with the 3270-style data stream. Each DBCS character string is surrounded by escape codes called shift out/shift in, or SO/SI. These codes originated from the need for the old-style printers to shift out from the EBCDIC character set, to the DBCS character set. The major manufacturers have different encodings for SO/SI; some manufacturers pad DBCS code with one byte of shift code information while others pad the DBCS code with two bytes of shift code information. These differences can cause problems in reading DBCS information about mainframes.

PCs, minicomputers, and workstations do not have SO/SI but have their own types of DBCS encodings that differ from manufacturer to manufacturer. SAS has several formats and informats that can read DBCS on SO/SI systems:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Language Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$KANJI</td>
<td>informat</td>
<td>Removes SO/SI from Japanese kanji DBCS</td>
</tr>
<tr>
<td>$KANJIX</td>
<td>informat</td>
<td>Adds SO/SI to Japanese kanji DBCS</td>
</tr>
<tr>
<td>$KANJII</td>
<td>format</td>
<td>Adds SO/SI to Japanese kanji DBCS</td>
</tr>
<tr>
<td>$KANJIX</td>
<td>format</td>
<td>Removes SO/SI from Japanese kanji DBCS</td>
</tr>
</tbody>
</table>
SAS Data Conversion between DBCS Encodings

Normally, DBCS data that is generated on one computer system is incompatible with data generated on another computer system. SAS has features that allow conversion from one DBCS source to another, as shown in the following table.

Table 5.2 DBCS Conversions

<table>
<thead>
<tr>
<th>Language Element</th>
<th>Type</th>
<th>Use</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCVT</td>
<td>function</td>
<td>Converts DBCS data from one operating environment to another</td>
<td>“KCVT Function” (p. 345)</td>
</tr>
<tr>
<td>CPORT</td>
<td>procedure</td>
<td>Moves files from one environment to another</td>
<td>Base SAS Procedures Guide</td>
</tr>
<tr>
<td>CIMPORT</td>
<td>procedure</td>
<td>Imports a transport file created by CPORT</td>
<td>Base SAS Procedures Guide</td>
</tr>
</tbody>
</table>

Avoiding Problems with Split DBCS Character Strings

- When working with DBCS characters, review your data to make sure that SAS recognizes the entire character string when data is imported or converted or used in a DATA step or a PROC step.
- On mainframe systems that use shift out/shift in escape codes, DBCS character strings can become truncated during conversion across operating environments.
- There is a possibility that DBCS character strings can be split when working with the PRINT, REPORT, TABULATE, and FREQ procedures. If undesirable splitting occurs, you might have to add spaces on either side of your DBCS character string to force the split to occur in a better place. The SPLIT= option can also be used with PROC REPORT and PROC PRINT to force string splitting in a better location.
Chapter 6

Time Zones

Overview of SAS Time Zone IDs

SAS must process local and international time and date values to accommodate international customers. This chapter explains time zones and how SAS processes the following:

- time zones
- Daylight Saving Time (DST)
- local time
- SAS datetime values

Specifying Time Zones in SAS

Definitions

Daylight Saving Time (DST)

Also referred to as *Summer Time*, *Daylight Saving Time (DST)* is a way of making better use of the daylight in the evenings by setting the clocks forward one hour during the longer days of summer and back again in the fall. DST begins in the northern hemisphere between March–April and ends between September–November.

Standard time begins in the northern hemisphere between September–November and ends between March–April. DST begins in the southern hemisphere between September–November and ends between March–April. Standard time begins in the southern hemisphere between March–April and ends between September–November.
Many of the countries in the northern hemisphere observe DST. DST is not a standard for all countries.

Local time
Specify the local time when a time zone is not specified by the TIMEZONE= system option.

Time zone
An area of the earth’s surface in which an identical and standard time is used. It is usually referred to as the local time. Time zones have political and geographical boundaries and might be adjusted for the convenience of local populations. Some geographically large countries, such as India and China, use only one time zone, but other large countries, such as Russia and the United States, have more than one time zone.

Time zone ID
Specifies a region and area separated by a forward slash (/). Asia/Tokyo is a time zone ID. Time zone IDs are compatible with Java time zones. For a list of time zone IDs, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819.

Time zone name
Specifies three or four characters that indicate a time zone. For a list of time zone names, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819.

Time zone offset
Specifies the number of hours and minutes that a time zone is off from the Universal Coordinated Time (UTC) in the form +|-hh:mm or +|-hhmm.

User local time
Specify the local time for the time zone that is specified by the TIMEZONE= system option.

Universal Coordinated Time (UTC)
Specify the time at the zero meridian, near Greenwich, England. UTC is a datetime value that uses the ISO 8601 basic form yyyymmddThhmmss+|-hhmm or the ISO 8601 extended form yyyy-mm-ddThh:mm:ss+|-hh:mm.

About Time Zones in SAS
SAS datetime values are measured in seconds, beginning with 01Jan1960 00:00:00 local time. For example, the SAS datetime value for 01Jan1960 00:00:00 in England is 0. In Japan, the SAS datetime value for 01Jan1960 00:00:00 is also 0. There is a nine-hour difference between England and Japan. At the same point in time, the datetime in England and the datetime in Japan cannot be 0. In order to work with absolute time and datetime values, SAS supports the UTC date, time, and datetime values in time zones.

You can name a time zone either by specifying a time zone ID or a time zone name. Time zone IDs name a region and an area in the form region/area. An example of a time zone ID is America/New_York. A time zone name specifies the time zone. An example of a time zone name is EST for Eastern Standard Time. When a specific time zone changes the time for DST or summer time, a separate time zone name is available for that alternate time. PST is Pacific Standard Time. PDT is Pacific Daylight Time. EET is Eastern European Time. EEST is Eastern European Summer Time. When you specify a time zone ID, SAS determines the time using DST. You do not need to determine whether the region/area uses alternate times. For a list of time zone IDs and time zone names, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819.

When you specify a time zone, SAS adds a time-zone-specific timestamp to data sets and SAS catalogs when the data set or catalog is created or modified. The time-zone-
specific timestamp is also added to the SAS log when SAS starts and to the output at execution. The SAS timestamp constant enables you to specify a timestamp using the ISO 8601 standard. Here is a timestamp using the SAS timestamp constant:

\[ t\text{stamp}='2013-05-17T09:15:30–05:00'dt; \]

When you specify a SAS timestamp constant, SAS converts the timestamp to a local datetime value by using the current time zone and the time zone offset.

Time zone settings affect the \( \text{TIME}( ) \) function, the \( \text{TODAY}( ) \) function, the \( \text{DATE}( ) \) function, and the \( \text{DATETIME}( ) \) function. When a time zone is set, the date and datetime values use the current time zone.

**Tasks and SAS Language Elements for Time Zone Processing**

**Set a Time Zone**

To set a time zone, use the TIMEZONE= system option:

\[ \text{options timezone='asia/tokyo';} \]

Use time zone names or time zone IDs for the value of the option. Enclose the value in quotation marks. For time zone names and time zone IDs, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819. For more information, see “TIMEZONE= System Option” on page 604.

**Determine a Time Zone ID Offset**

You use the TZONEOFF( ) function to determine a time zone name or time zone ID offset:

- The TZONEOFF( ) function returns the time zone offset for the current time zone.
- The TZONEOFF('time-zone-id') function returns the time zone offset for the 'time-zone-id'.

This program returns the time zone offset for the current time zone (EST) and for Tokyo:

\[ \begin{align*}
\text{data _null_;} \\
\quad \text{o1=tzoneoff();} \\
\quad \text{o2=tzoneoff('asia/tokyo');} \\
\quad \text{put o1 time.;} \\
\quad \text{put o2 time.;} \\
\text{run;}
\end{align*} \]

Here is the output in the SAS log:

\[-5:00:00\]
\[9:00:00\]

To find the difference between two time zones, you can use the ABS( ) function:

\[ \text{diff=abs(tzoneoff('america/new_york') - tzoneoff('asia/tokyo'));} \]

For more information, see “TZONEOFF Function” on page 405.

**Determine a Time Zone ID or Time Zone Name**

Use these functions to determine a time zone name or time zone ID:

- The TZONEID( ) function returns the current time zone ID.
The TZONENAME( ) function returns the current time zone name based on a time zone ID and a SAS datetime value or based on a SAS datetime value only.

Here are some examples of using these functions. March 10 of 2013 is the first day of DST:

```sas
options timezone='America/Chicago';
data _null_;  
tzid=tzoneid();  
put 'Current time zone is ' tzid;  
tzn=tzonename('america/los_angeles');  
put 'Time zone for Los Angeles: ' tzn;  
tznST=tzonename('america/los_angeles','10mar2013:01:00:00'dt);  
put 'Time zone for Los Angeles standard time: ' tznST;  
tznDT=tzonename('america/los_angeles','10mar2013:02:00:00'dt);  
put 'Time zone for Los Angeles daylight time: ' tznDT;  
tznSDT=tzonename('10mar2013:02:00:00'dt);  
put 'Time zone name for this SAS datetime: ' tznSDT;  
run;
```

Here is the output:

```
Current time zone is AMERICA/CHICAGO
Time zone for Los Angeles: PST
Time zone for Los Angeles standard time: PST
Time zone for Los Angeles daylight time: PDT
Time zone name for this SAS datetime: CDT
```

For more information, see “TZONENAMEID Function” on page 402 and “TZONENAME Function” on page 404.

### Convert Datetime Values between SAS and UTC

These functions convert SAS datetime values to UTC and UTC to SAS datetime values:

- The TZONE2U( ) converts a SAS datetime value to a UTC datetime value.
- The TZONEU2S( ) function converts a UTC datetime value to a SAS datetime value.

Here are some examples of using these functions:

```sas
options timezone='est';
data _null_;  
put ' The time zone is EST';  
diff=abs(tzoneoff('america/new_york') - tzoneoff('europe/london'));  
put ' New York-London difference: ' diff time.;  
diff=abs(tzoneoff('america/new_york') - tzoneoff('asia/tokyo'));  
put ' New York-Tokyo difference: ' diff time.;  
put ' The SAS datetime is 2013-03-15T09:15:00+00:00 ';  
put ' Change a SAS datetime to a UTC value ';  
put ' ';  
put ' The time zone offset +00:00 is for London ';  
put ' Subtract the 5 hours for the EST time zone offset';  
stu1=tzones2u('2013-03-15T09:15:00+00:00'dt);  
put ' STU1 Using E8601DX:' stu1 e8601dx.;  
put ' ';  
put ' 2013-03-15 9:15 AM in Tokyo is 2013-03-14 7:15 PM in New York';  
put ' Subtract the 5 hours for the EST time zone offset';
```
stu2=tzones2u('2013-03-15T09:15:00+00:00'dt, 'Asia/Tokyo');
put ' STU2 Using E8601DX:' stu2 e8601dx.;
put ' Change a UTC to a SAS datetime value. ';
put ' +00:00 is the time zone offset for London.';
put ' Subtract the 5 hours for the EST time zone offset';
uts1=tzoneu2s('2013-03-15T09:15:00+00:00'dt);
put ' UTS1 Using DATETIME:' uts1 datetime.;
put ' 9:15:00+00:00 is 18:15:00 in Tokyo. ';
put ' Subtract the 5 hours for the EST time zone offset';
uts2=tzoneu2s('2013-03-15T09:15:00+00:00'dt, 'Asia/Tokyo');
put ' UTS2 Using DATETIME:' uts2 datetime.;
run;

Here is the output:
The time zone is EST
   New York-London difference: 5:00:00
   New York-Tokyo difference: 14:00:00
   The SAS datetime is 2013-03-15T09:15:00+00:00

Change a SAS datetime to a UTC value

   The time zone offset +00:00 is for London
   Subtract the 5 hours for the EST time zone offset
   STU1 Using E8601DX: 2013-03-15T04:15:00-05:00
2013-03-15 9:15 AM in Tokyo is 2013-03-14 7:15 PM in New York
   Subtract the 5 hours for the EST time zone offset
   STU2 Using E8601DX: 2013-03-14T14:15:00-05:00

Change a UTC to a SAS datetime value.

   +00:00 is the time zone offset for London.
   Subtract the 5 hours for the EST time zone offset
   UTS1 Using DATETIME: 2013MAR13:23:15:00
   9:15:00+00:00 is 18:15:00 in Tokyo.
   Subtract the 5 hours for the EST time zone offset
   UTS2 Using DATETIME: 2013MAR13:13:15:00

For more information, see “TZONES2U Function” on page 406.

Write ISO 8601 Time and Datetime Values Based on the Time Zone
These formats write ISO 8601 time and datetime values using basic and extended notation. The time, the UTC offset, or both are based on the value of the TIMEZONE= system option:

• The B8601TX. and E8601TX. formats convert a UTC time value to a user local time and write the time value with a time zone offset.
• The B8601DX. and E8601DX. formats convert a UTC datetime value to a user local time and write the datetime value with a time zone offset.
• The B8601LX. and E8601LX. formats write a local datetime value using the UTC offset for the user local time zone.
This program formats time and datetime values based on the time zone ID Australia/Sydney:

```sas
options timezone='Australia/Sydney';
data _null_;  
st='18:33:40't;  
sdt='2013-03-17T14:30:22+00:00'dt;  
put 'Time B8601TX:' st b8601tx.;  
put 'Time E8601TX:' st e8601tx.;  
put 'UTC B8601DX:' sdt b8601dx.;  
put 'UTC E8601DX:' sdt e8601dx.;  
put 'UTC B8601LX:' sdt b8601lx.;  
put 'UTC E8601LX:' sdt e8601lx.;  
run;
```

Here is the output:

```
Time B8601TX:   04:33:40+10:00  
Time E8601TX:04:33:40+10:00  
UTC B8601DX:      20130318T123022+1100  
UTC E8601DX: 2013-03-18T12:30:22+11:00  
UTC B8601LX:      20130318T013022+1100  
UTC E8601LX: 2013-03-18T01:30:22+11:00
```

For more information, see “B8601DXw. Format” on page 99, “B8601LXw. Format” on page 111, and “E8601TXw. Format” on page 113.

**Write SAS Datetime Values Using a Time Zone Offset**

These formats write SAS datetime values using a UTC offset based on the value of the `TIMEZONE=` system option:

- The NLDATMZ. format writes a SAS datetime value using the form `ddmmmyyyy:hh:mm:ss +|-hhmm`.
- The NLDATMTZ. format writes a SAS time value using the form `hh:mm:ss +|-hhmm`.
- The NLDATMWZ. format writes a SAS datetime value as the day of the week, the month, the day, the year, and AM | PM in the form `day-of-week, month-name dd, yyyy AM|PM +hhmm`.

This program formats SAS time and datetime values using a time zone offset:

```sas
options timezone='Indian/Maldives';
data _null_;  
st='18:33:40't;  
sdt='2013-03-17T14:30:22+00:00'dt;  
put 'Time NLDATMTZ:' st nldatmtz.;  
put 'SAS datetime NLDATMZ:' sdt nldatmz.;  
put 'SAS datetime NLDATMWZ:' sdt nldatmwz.;  
run;
```

Here is the output:

```
Time NLDATMTZ:18:33:40 +0500  
SAS datetime NLDATMZ:17Mar2013:19:30:22 +0500  
SAS datetime NLDATMWZ:Sunday, March 17, 2013 07:30:22 AM +0500
```

For more information, see “NLDATMTZw. Format” on page 162, “NLDATMZw. Format” on page 175, and “NLDATMWZw. Format” on page 165.
Time Zone Example

This example shows the arrival time of a flight from Los Angeles to Tokyo:

```sas
/* Set the time zone */
options timezone='America/Los_Angeles';
data depart;
/* Set the departure time */
depart='2013-05-17T09:15:00-08:00'dt;
put 'Depart Los Angeles: ' depart nldatmwz.;
/* Set the flight time */
ftime='13:00't;
put 'Flight time=' ftime time.;
utc=depart+ftime;
put 'Arrive PST=' utc nldatmwz.;
put 'Arrive UTC=' utc nldatmwz.;
run;
/* Set the time zone for Tokyo */
options timezone='Asia/Tokyo';
data arrive;
set depart;
put 'Arrive in Tokyo ' utc nldatmwz.;
run;
```

Here is the output:

```
Depart Los Angeles: Friday, May 17, 2013 10:15:00 AM -0700
Flight time=13:00:00
Arrive PST=Friday, May 17, 2013 11:15:00 PM -0700
Arrive UTC=Friday, May 17, 2013 11:15:00 PM -0700
NOTE: The data set WORK.DEPART has 1 observations and 3 variables.
NOTE: DATA statement used (Total process time):
   real time           0.01 seconds
   cpu time            0.01 seconds
```

```sas
/* Set the time zone */
options timezone='America/Los_Angeles';
data depart;
/* Set the departure time */
depart='2013-05-17T09:15:00-08:00'dt;
put 'Depart Los Angeles: ' depart nldatmwz.;
/* Set the flight time */
ftime='13:00't;
put 'Flight time=' ftime time.;
utc=depart+ftime;
put 'Arrive PST=' utc nldatmwz.;
put 'Arrive UTC=' utc nldatmwz.;
run;
/* Set the time zone for Tokyo */
options timezone='Asia/Tokyo';
data arrive;
set depart;
put 'Arrive in Tokyo ' utc nldatmwz.;
run;
```
NOTE: There were 1 observations read from the data set WORK.DEPART.
NOTE: The data set WORK.ARRIVE has 1 observations and 3 variables.
NOTE: DATA statement used (Total process time):
    real time          0.01 seconds
    cpu time           0.01 seconds
Part 2

Autocall Macros for NLS

Chapter 7

Dictionary of Autocall Macros for NLS ........................................... 51
Chapter 7

Dictionary of Autocall Macros for NLS

Autocall Macro Entries by Category

The following table provides brief descriptions of the SAS NLS autocall macros. For more detailed descriptions, see the NLS entry for each macro.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCS</td>
<td>%KLOWCASE and %QKLOWCAS Autocall Macros (p. 51)</td>
<td>Change uppercase characters to lowercase.</td>
</tr>
<tr>
<td></td>
<td>%KTRIM and %QKTRIM Autocall Macros (p. 52)</td>
<td>Trim trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%KVERIFY Autocall Macro (p. 52)</td>
<td>Returns the position of the first character unique to an expression.</td>
</tr>
</tbody>
</table>

Dictionary

%KLOWCASE and %QKLOWCAS Autocall Macros

Change uppercase characters to lowercase.

Category: DBCS

Requirement: MAUTOSOURCE system option
%KLOWCASE (text | text expression)

%QKLOWCAS (text | text expression)

Details

The %KLOWCASE and %QKLOWCAS macros change uppercase alphabetic
characters to their lowercase equivalents. If the argument might contain a special
character or mnemonic operator, listed below, use %QKLOWCAS.

%KLOWCASE returns a result without quotation marks, even if the argument has
quotation marks. %QKLOWCAS produces a result with the following special characters
and mnemonic operators masked so that the macro processor interprets them as text
instead of as elements of the macro language:

& % ' " ( ) + - * / < > = ¬ ^ ~ ; , blank AND OR NOT EQ NE LE LT GE GT IN

Autocall macros are included in a SAS library. This library might not be installed at your
site or might be a site-specific version. If you cannot access this macro or if you want to
find out if the library is a site-specific version, see your on-site SAS support personnel.

%KTRIM and %QKTRIM Autocall Macros

Trim trailing blanks.

Category: DBCS
Requirement: MAUTOSOURCE system option

Syntax

%KTRIM (text | text expression)

%QKTRIM (text | text expression)

Details

The KTRIM macro and the QKTRIM macro trim trailing blanks. If the argument
contains a special character or mnemonic operator, listed below, use %QKTRIM.

QKTRIM produces a result with the following special characters and mnemonic
operators masked so that the macro processor interprets them as text instead of as elements of the macro language:

& % ' " ( ) + - * / < > = ¬ ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN

Autocall macros are included in a SAS library. This library might not be installed at your
site or might be a site-specific version. If you cannot access this macro or if you want to
find out if the library is a site-specific version, see your on-site SAS support personnel.

%KVERIFY Autocall Macro

Returns the position of the first character unique to an expression.

Category: DBCS
Requirement: MAUTOSOURCE system option
Syntax

%KVERIFY (source, excerpt)

Syntax

source
is text or a text expression that you want to examine for characters that do not exist in excerpt.

excerpt
is text or a text expression that defines the set of characters that %KVERIFY uses to examine source.

Details

%KVERIFY returns the position of the first character in source that is not also present in excerpt. If all characters in source are present in excerpt, %KVERIFY returns 0.

Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel.
Part 3

Data Set Options for NLS

Chapter 8
Dictionary of Data Set Options for NLS .......................... 57
Chapter 8
Dictionary of Data Set Options for NLS

Data Set Options by Category

NLS affects the data set control category of options for selected data set options. The following table provides brief descriptions of the data set options. For more detailed descriptions, see the dictionary entry for each data set option:

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Control</td>
<td>ENCODING= Data Set Option</td>
<td>Overrides the encoding to use for reading or writing a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>(p. 57)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OUTREP= Data Set Option</td>
<td>Specifies the data representation for the output SAS data set.</td>
</tr>
<tr>
<td></td>
<td>(p. 60)</td>
<td></td>
</tr>
</tbody>
</table>

Dictionary

ENCODING= Data Set Option

Overrides the encoding to use for reading or writing a SAS data set.

Valid in: DATA step and PROC steps

Category: Data Set Control

Syntax

ENCODING= ANY | ASCIIANY | EBCDICANY | encoding-value
Syntax Description

ANY
specifies that no transcoding occurs.

Note  ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

ASCIIANY
specifies that no transcoding occurs when the mixed encodings are ASCII encodings.

EBCDICANY
specifies that no transcoding occurs when the mixed encodings are EBCDIC encodings.

encoding-value
specifies an encoding value.

See  Chapter 3, “Encoding for NLS,” on page 9

Details

The value for ENCODING= indicates that the SAS data set has a different encoding from the current session encoding. When you read data from a data set, SAS transcodes the data from the specified encoding to the session encoding. When you write data to a data set, SAS transcodes the data from the session encoding to the specified encoding.

Input Processing

By default, encoding for input processing is determined as follows:

• If the session encoding and the encoding that is specified in the file are different, SAS transcodes the data to the session encoding.

• If a file has no encoding specified, but the file's data representation is different from the encoding of the current session, then SAS transcodes the data to the current session.

Output Processing

By default, encoding for output processing is determined as follows:

• Data is written to a file using the encoding of the current session, except when a different output representation is specified using the OUTREP= data set option, the OUTENCODING= option in the LIBNAME statement, or the ENCODING= data set option.

• If a new file replaces an existing file, then the new file inherits the encoding of the existing file.

• If an existing file is replaced by a new file that was created under a different operating environment or that has no encoding specified, the new file uses the encoding of the current session.

Note: Character metadata and data output appears garbled if you specify a different encoding from where the data set was created. In this example, the data set to be printed is internally encoded as ASCII. However, the data set option specifies an EBCDIC encoding. SAS attempts to transcode the data from EBCDIC to ASCII, but the data is already in ASCII. The result is garbled data.

data a;
x=1;
abc='abc';
run;
proc print data=a (encoding="ebcdic");
run;

Note: The following values for ENCODING= are invalid:

- UCS2
- UCS4
- UTF16
- UTF32

Comparisons

- Session encoding is specified using the ENCODING= system option or the LOCALE= system option, with each operating environment having a default encoding.
- You can specify encoding for a SAS library by using the LIBNAME statement's INENCODING= option (for input files) and the OUTENCODING= option (for output files). If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.

Examples

Example 1: Creating a SAS Data Set with Mixed Encodings and with Transcoding Suppressed

By specifying the data set option ENCODING=ANY, you can create a SAS data set that contains mixed encodings, and suppress transcoding for either input or output processing.

In this example, the new data set MYFILES.MIXED contains some data that uses the Latin1 encoding, and some data that uses the Latin2 encoding. When the data set is processed, no transcoding occurs. For example, the correct Latin1 characters in a Latin1 session encoding and correct Latin2 characters in a Latin2 session encoding are displayed.

```sas
libname myfiles 'SAS data-library';
data myfiles.mixed (encoding=any);
  set work.latin1;
  set work.latin2;
run;
```

Example 2: Creating a SAS Data Set with a Particular Encoding

For output processing, you can override the current session encoding. This action might be necessary, for example, if the normal access to the file uses a different session encoding.

For example, if the current session encoding is Wlatin1, you can specify ENCODING=WLATIN2 in order to create the data set that uses the encoding WLatin2. The following statements tell SAS to write the data to the new data set using the WLatin2 encoding instead of the session encoding. The encoding is also specified in the descriptor portion of the file.

```sas
libname myfiles 'SAS data-library';
data myfiles.difencoding (encoding=wlatin2);
```
Example 3: Overriding Encoding for Input Processing

For input processing, you can override the encoding that is specified in the file, and specify a different encoding.

For this example, the current session encoding is EBCDIC-870, but the file has the encoding value EBCDIC-1047 in the descriptor information. By specifying ENCODING=EBCDIC-870, SAS does not transcode the data, but instead displays the data using EBCDIC-870 encoding.

```
proc print data=myfiles.mixed (encoding=ebcidic870);
run;
```

See Also

- Conceptual discussion in Chapter 3, “Encoding for NLS,” on page 9
- Options in Statements and Commands:
  - “ENCODING= Option” on page 630
  - “INENCODING= and OUTENCODING= Options” on page 634
- System Options:
  - “ENCODING System Option: UNIX, Windows, and z/OS” on page 587
  - “LOCALE System Option” on page 591

OUTREP= Data Set Option

Specifies the data representation for the output SAS data set.

Valid in: DATA step and PROC steps
Category: Data Set Control
See: “OUTREP= Data Set Option” in SAS Data Set Options: Reference.

Syntax

```
OUTREP= format
```
Part 4

Formats for NLS

Chapter 9
Overview to NLS Formats ..................................................... 63

Chapter 10
Dictionary of Formats for NLS ............................................. 83
Chapter 9
Overview to NLS Formats

International Date and Datetime Formats

SAS supports international formats that are equivalent to some of the most commonly used English-language date formats. In each case the format works like the corresponding English-language format. Only the maximum, minimum, and default widths are different.

Table 9.1 International Date and Datetime Formats

<table>
<thead>
<tr>
<th>Language</th>
<th>English Format</th>
<th>International Format</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATEN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>English</td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td></td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td></td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>WEEKDAY.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>105</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATIM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEN.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATIM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEN.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATIM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEN.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATIM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENW.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>WEEKDDATX.</td>
<td>EURDFWDX.</td>
<td>1</td>
<td>32</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WORDDATX.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATM.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEWN.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATE.</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATM.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEWN.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATM.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEWN.</td>
<td>6</td>
<td>200</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NLDATEWX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>16</td>
</tr>
</tbody>
</table>
## Currency Representation

### Overview to Currency

Currency is the medium of exchange, which is specific to a country. SAS provides formats and informats for reading and writing currency.

### U.S. Dollars

The DOLLARw.d formats and informats were first introduced to read and write American currency. DOLLARw.d

- uses the dollar sign ($) currency symbol to precede U.S. currency
- uses a comma (,) as the thousands separator and a dot (.) as the decimal separator.

Example:

$12,345.00

DOLLARXw.d also writes currency with a leading dollar sign ($), but uses a dot (.) as the thousands separator and a comma (,) as the decimal separator. The reversal of the dot and comma for currency formatting is a convention used in many European countries.

Example:

$12.345,00

Limitations of the DOLLAR formats and informats are:

- the lack of support for all currency symbols
- the reversal of the dot and comma for currency formatting is not used by all European countries.
- the appearance of the currency symbol varies by computer (an EBCDIC-based computer and an ASCII-based computer render characters differently).

### Euros

The EUROw.d formats and informats were introduced to support the euro currency that was established by the European Monetary Union (EMU), which was formed in 1999. EUROw.d

Example:

```plaintext
options locale=English_UnitedKingdom;
data _null_;`
x=12345;
put x euro10.2;
run;

Output:
€12.345,00

Limitations of the EURO formats and informats are:

- uses the letter E as currency symbol if the € symbol is not supported by the session encoding.
- Euro format is not locale sensitive. It uses a comma(,) as the thousands separator and a dot(.) as the decimal separator.

**Customized Currency Representations**

To create a customized currency representation, you can use the FORMAT procedure. The following example shows the creation of unique formats for the Australian dollar, the Swiss franc, and the British pound. For details about the FORMAT procedure, see Base SAS Procedures Guide.

**Example Code 9.1 SAS Code That Customizes Currency Representations**

```sas
proc format;
  picture aud low-<0='0,000,000,009.00'
       (prefix='-AU$' mult=100)
    0–high='0,000,00,009.00 '
       (prefix='AU$' mult=100);

  picture sfr low-<0='0,000,000,009.00'
       (prefix='-SFr.' mult=100)
    0–high='0,000,00,009.00 '
       (prefix='-SFr.' mult=100);

  picture bpd low-<0='0,000,000,009.00'
       (prefix='-BPd.' mult=100)
    0–high='0,000,00,009.00 '
       (prefix='BPd.' mult=100);
run;
```

```sas
data currency;
  input aud sfr bpd 12.2;
datalines;
  12345 12345 12345
  0 0 0
  -12345 -12345 -12345
;```

```sas
proc print data=currency noobs;
  var aud sfr bpd;
  format aud aud. sfr sfr. bpd bpd. ;
  title 'Unique Currency Formats';
run;
```
Customizing currency representations offers flexibility, but requires a programming solution.

**Localized National and International Currency Representations**

The NLMNYw.d and NLMNYw.d formats and informats were introduced to represent localized currency in two forms:

Localized national currency representation reflects the customs and conventions of the locale. National formats are specified using the NLMNYw.d formats and informats. You must also use the LOCALE= option to specify the locale when using the NLMNYw.d formats and informats.

Example: `options locale=english_UnitedStates; data _null_; x=12345; put x nlmny15.2; run;`

Output:

$12,345.00

Selected national currency representations follow:

<table>
<thead>
<tr>
<th>LOCALE</th>
<th>Currency</th>
<th>National Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English_UnitedStates</td>
<td>U.S. dollars</td>
<td>$12,345.00</td>
</tr>
<tr>
<td>French_Canada</td>
<td>Canadian dollars</td>
<td>12 345,00 $</td>
</tr>
<tr>
<td>French_France</td>
<td>Euros</td>
<td>12 345,00 €</td>
</tr>
<tr>
<td>French_Switzerland</td>
<td>Swiss francs</td>
<td>SFr. 12’345.00</td>
</tr>
<tr>
<td>German_Germany</td>
<td>Euros</td>
<td>12.345,00 €</td>
</tr>
<tr>
<td>German_Luxembourg</td>
<td>Euros</td>
<td>12.345,00 €</td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>Euros</td>
<td>12.345,00 €</td>
</tr>
<tr>
<td>Spanish_Venezuela</td>
<td>Venezuelan bolivars</td>
<td>VEF12.345,00</td>
</tr>
</tbody>
</table>

The localized renderings show the native customs for representing currency. For example, although these selected EMU countries might use the same euro currency, their depiction of the currency varies. Whereas French_France uses no thousands separator but uses a comma as a decimal separator, German_Germany and Spanish_Spain use a dot as a thousands separator and a comma as a decimal separator.
Localized International currency representation conforms to ISO standard 4217. International forms are specified using the NLMDYw.d formats and informats. International forms are commonly used to show a comparison of world currencies, for example, airline ticket, trade, and stock market pricing. You must also use the LOCALE= option to specify the locale when using the NLMDYw.d formats and informats. The letter “I,” which signifies “International,” is appended to the format and informat names.

Example:

```plaintext
options locale=english_UnitedStates;
data _null_;  
x=12345; put x nlmnyi15.2;  
run;
```

Output:

```
USD12,345.00
```

Selected international currency representations follow:

<table>
<thead>
<tr>
<th>LOCALE=</th>
<th>Currency</th>
<th>International Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English_UnitedStates</td>
<td>U.S. dollars</td>
<td>USD12,345.00</td>
</tr>
<tr>
<td>French_Canada</td>
<td>Canadian dollars</td>
<td>12 345,00 CAD</td>
</tr>
<tr>
<td>French_France</td>
<td>Euros</td>
<td>12 345,00 EUR</td>
</tr>
<tr>
<td>French_Luxembourg</td>
<td>Euros</td>
<td>12.345,00 EUR</td>
</tr>
<tr>
<td>German_Germany</td>
<td>Euros</td>
<td>12.345,00 EUR</td>
</tr>
<tr>
<td>German_Switzerland</td>
<td>Swiss francs</td>
<td>CHF12,345.00</td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>Euros</td>
<td>12.345,00 EUR</td>
</tr>
<tr>
<td>Spanish_Venezuela</td>
<td>Venezuelan bolivars</td>
<td>VEF12.345,00</td>
</tr>
</tbody>
</table>

The international renderings also reflect native customs for representing currency. For example, although all locales use a comma as the thousands separator and a dot as the decimal separator, they vary the placement of the ISO currency code. Whereas the EMU countries put the currency code after the currency, English_UnitedStates, German_Switzerland, and Spanish_Venezuela precede the currency with the ISO code.

For a complete list of the ISO standard 4217 currency codes, see http://www.xe.com/iso4217.php.

A primary limitation of using localized national and international currency representations is their dependence on a value for the LOCALE= system option.
**Unique National and International Monetary Representations**

The NLMNLISOw.d and NLMNIISOw.d formats and informats were introduced to uniquely represent each currency without having to also use the LOCALE= option to specify the locale. Each currency is specified by a unique ISO standard 4217 currency code.

Unique national monetary representation

is specified by the unique ISO currency code. National formats are specified using the NLMNLISOw.d formats and informats. In the following example, USD is the ISO currency code for American dollars.

*Note:* When using the NLMNLISOw.d formats and informats, the LOCALE= option is optional.

Example: `data _null_; x=12345; put x nlmnlusd15.2; run;`

Output:

**US$12,345.00**

Selected unique national currency representations follow:

In this example, locale is specified as fr_FR.

`data _null_; x=12345; put x nlmnleur15.2; run;`

Output:

**€12,345.00**

**Table 9.4  Unique Currency Representations by ISO Currency Code**

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Currency</th>
<th>National Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD</td>
<td>U.S. dollars</td>
<td>US$12,345.00</td>
</tr>
<tr>
<td>CAD</td>
<td>Canadian dollars</td>
<td>CA$12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>CHF</td>
<td>Swiss francs</td>
<td>SFr.12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>€12,345.00</td>
</tr>
<tr>
<td>VEB</td>
<td>Venezuelan bolivars</td>
<td>BsF.12,345.00</td>
</tr>
</tbody>
</table>

A currency symbol or a currency code precedes most currencies. Also used are a comma as the thousands separator and a dot as the decimal separator. If the currency symbol of the local currency is not supported in the current SAS session encoding, the NLMNLxxxw.d format formats the value with the 3-letter ISO currency code.

Unique international monetary representation

is specified by the unique ISO currency code. International formats are specified using the NLMNIISOw.d formats and informats. International forms are commonly
used to show a comparison of world currencies, for example, airline ticket, trade, and stock market pricing. The letter “I”, which signifies “International”, is appended to the format and informat names. In the following example, USD is the ISO currency code for American dollars.

*Note:* When using the NLMNI/ISOw.d formats and informats, you do not use the LOCALE= option to specify the locale.

Example: data _null_; x=12345; put x nlmniusd15.2; run; Output: USD12,345.00

Selected international currency representations follow:

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Currency</th>
<th>International Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD</td>
<td>U.S. dollars</td>
<td>USD12,345.00</td>
</tr>
<tr>
<td>CAD</td>
<td>Canadian dollars</td>
<td>CAD12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>CHF</td>
<td>Swiss francs</td>
<td>CHF12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>VEB</td>
<td>Venezuelan bolivars</td>
<td>Not found</td>
</tr>
</tbody>
</table>

The international renderings precede the currency with the appropriate ISO code. Also used are a comma as the thousands separator and a dot as the decimal separator.

**Example: Representing Currency in National and International Formats**

This SAS program uses the exchange rates for selected Asia-Pacific countries against the U.S. dollar. In the output, each country's currency is represented using a national format and an international format.

**Example Code 9.2  SAS Code That Formats National and International Currency Formats**

data curr;
input ex_date mmddyy. usd aud hkd jpy sgd 12.2;
datalines;
061704 1.00000 1.45349 7.79930 110.110 1.71900 1
These exchange rates, which were effective June 17, 2004, are specified as data in the SAS program.

These NLMNL/ISO formats are applied to each of the numeric data items that are specified in the INPUT statement. These formats show currencies in the appropriate national formats.

These NLMNI/ISO formats are applied to each of the numeric data items that are specified in the INPUT statement. These formats show currencies in the appropriate international formats.
European Currency Conversion

Overview to European Currency Conversion

SAS enables you to convert European currency from one country's currency to an equivalent amount in another country's currency. You can also convert a country's currency to euros, and you can convert euros to a specific country's currency.

SAS provides a group of formats, informats, and a function to use for currency conversion. The set of formats EURFR/ISO can be used to convert specific European currencies to an amount in euros. ISO represents an ISO standard 4214 currency code. For a complete list of the ISO standard 4217 currency codes, see http://www.xe.com/iso4217.php.

Fixed Rates for Euro Conversion

Twenty-seven European countries comprise the EMU (European Monetary Union). The conversion rates for 17 countries are fixed, and are incorporated into the EURFR/ISO and EURTO/ISO formats and into the EUROCURR function. The following table lists the currency codes and conversion rates for the specific currencies whose rates are fixed.

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Conversion Rate</th>
<th>Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATS</td>
<td>13.7603</td>
<td>Austrian schilling</td>
</tr>
<tr>
<td>BEF</td>
<td>40.3399</td>
<td>Belgian franc</td>
</tr>
<tr>
<td>CYP</td>
<td>0.585274</td>
<td>Cyprus pound</td>
</tr>
<tr>
<td>DEM</td>
<td>1.95583</td>
<td>Deutsche mark</td>
</tr>
<tr>
<td>ESP</td>
<td>166.386</td>
<td>Spanish peseta</td>
</tr>
<tr>
<td>EEK</td>
<td>15.6466</td>
<td>Estonian kroon</td>
</tr>
<tr>
<td>EUR</td>
<td>1</td>
<td>Euro</td>
</tr>
<tr>
<td>FIM</td>
<td>5.94573</td>
<td>Finnish markka</td>
</tr>
</tbody>
</table>
ISO Currency Code | Conversion Rate | Currency
--- | --- | ---
FRF | 6.55957 | French franc
GRD | 340.750 | Greek drachma
IEP | 0.787564 | Irish pound
ITL | 1936.27 | Italian lira
LUF | 40.3399 | Luxembourg franc
MTL | 0.429300 | Maltese lira
NLG | 2.20371 | Dutch guilder
PTE | 200.482 | Portuguese escudo
SIT | 239.640 | Slovenian tolar
SKK | 30.1260 | Slovak koruna

**Variable Rates for Euro Conversion**

For 13 countries in the EMU, currency conversion rates can fluctuate. The conversion rates for these countries are stored in an ASCII text file that you reference with the EURFRTBL fileref. For example, you can store the variable rates in a file named variableRates.txt, and reference the file with the `Filename EURFRTBL "variables.txt"` statement. The contents of variableRates.txt could be:

```
EURFRCHF=1.5260
EURFRPLZ=1.3650
```

You can convert Polish Zloty to euros with the following code:

```plaintext
data _null;
x=12345;
put x eurfrplz15.2;
run;
output:
€2.939,29
```

**Example: Converting between a European Currency and Euros**

The following example shows the conversion from Belgian francs to euros. The EURFRBEF format divides the country's currency amount by the exchange rate:

```
CurrencyAmount / ExchangeRate
12345 / 40.3399
```

**Example Code 9.3**  Example Code: Conversion from Belgian Francs to Euros
data _null_
  x=12345 /*convert from Belgian francs to euros*/
  put x eurfrbef15.2;
run;

Output:
  e306,02

The following example shows the conversion of euros to Belgian francs. The EUROTBEF format multiplies euros by the target currency's exchange rate:

\[
\text{EurosAmount} \times \text{ExchangeRate}
\]

12345 \times 40.3399

data _null_
  x=12345; /*convert from euros to Belgian francs*/
  put x eurtobef15.2;
run;

Output:
  497996.07

**Direct Conversion between European Currencies**

The EUROCURRE function uses the conversion rate tables to convert between currencies. For conversion between the currencies of two countries,

1. SAS converts the amount to euros.

   *Note:* SAS stores the intermediate value as precisely as the operating environment allows, and does not round the value.

2. SAS converts the amount in euros to an amount in the target currency.

   \[
   \text{SourceCurrencyAmount} \rightarrow \text{EurosAmount} \rightarrow \text{TargetCurrencyAmount}
   \]

BelgianFrancs \rightarrow euros

12345 / 40.3399 = 306.02456 euros

Euros \rightarrow FrenchFrancs

306.02456 \times 6.55957 = 2007.3895 French francs

data _null_
  x=eurocurr(12345,'bef','frf'); /*convert from Belgian francs to French francs*/
  put x=;
run;

Output:
  x=2007.389499

SAS converts Belgian francs to euros, and then euros to French francs.
Exceptions for Date and Time Default Widths

The length of the month-name and the week-name in some locale and encoding combinations might exceed the default width in some formats. Do not use the default width for these combinations. Refer to the following tables for the affected locales, encodings, and formats. Use the recommended widths for accurate output.

The following table lists the encoding and locale combinations where the localized output might exceed the default width.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Locale</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLATIN1</td>
<td>FR_FR, IT_IT, DE_DE, ES_ES, EN_US, EN_GB</td>
</tr>
<tr>
<td>WLATIN2</td>
<td>CS_CS, HU_HU, PL_PL</td>
</tr>
<tr>
<td>WCYRILLIC</td>
<td>RU_RU</td>
</tr>
<tr>
<td>EUC_CN</td>
<td>EN_US &amp; ZH_CN</td>
</tr>
<tr>
<td>SHIFT-JIS</td>
<td>EN_US JA_JP4-6</td>
</tr>
<tr>
<td>EUC-KR</td>
<td>EN_US KO_KR</td>
</tr>
<tr>
<td>BIG5</td>
<td>EN_US ZT_TW ZT_HK</td>
</tr>
<tr>
<td>UTF-8</td>
<td>all listed locales</td>
</tr>
</tbody>
</table>

The following table lists the recommended widths for encoding and locale combinations where the localized output might exceed the default width. You might need to use the recommended width in the specified formats.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Locale</th>
<th>Format</th>
<th>Default Width</th>
<th>Recommended Width (&gt;=)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLATIN1</td>
<td>EN_GB</td>
<td>nldatmw</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>WLATIN2</td>
<td>CS_CS</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>EUC-CN</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>ZH_CN</td>
<td>nldatmw</td>
<td>30</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>Encoding</td>
<td>Locale</td>
<td>Format</td>
<td>Default Width</td>
<td>Recommended Width (≥)</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>------------</td>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>SHIFT-JIS</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>JA_JP</td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>JA_JP</td>
<td>nldatmw</td>
<td>30</td>
<td>34</td>
</tr>
<tr>
<td>EUC-KR</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>KO_KR</td>
<td>nldatmw</td>
<td>30</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>KO_KR</td>
<td>nldatmwz</td>
<td>40</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>BIG5</td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>ZT_HK</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>ZT_TW</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>UTF-8</td>
<td>CS_CS</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>EN_GB</td>
<td>nldatmw</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>EN_US</td>
<td>nltimap</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>JA_JP</td>
<td>nldatmw</td>
<td>30</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>JA_JP</td>
<td>nldateyq</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>JA_JP</td>
<td>nldatmyq</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>JA_JP</td>
<td>nldatmwz</td>
<td>40</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>KO_KR</td>
<td>nldatew</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>KO_KR</td>
<td>nldatmw</td>
<td>30</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>KO_KR</td>
<td>nldatmwz</td>
<td>40</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>10</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>RU_RU</td>
<td>nldatew</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>RU_RU</td>
<td>nldatmw</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>ZH_CN</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>ZH_CN</td>
<td>nldatmw</td>
<td>30</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>ZH_CN</td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
</tr>
<tr>
<td>Encoding</td>
<td>Locale</td>
<td>Format</td>
<td>Default Width</td>
<td>Recommended Width (&gt;=)</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>ZT_HK</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmw</td>
<td>30</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>ZT_TW</td>
<td>nldatew</td>
<td>20</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmw</td>
<td>30</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nldatmwz</td>
<td>40</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nltimap</td>
<td>10</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 10

Dictionary of Formats for NLS

Categories of NLS Formats ................................................................. 86

Dictionary .............................................................. 99

B8601DXw. Format ................................................................. 99
B8601LXw. Format ................................................................. 101
B8601TXw. Format ................................................................. 102
BESTDOTXw. Format ............................................................. 104
$BIDIw. Format ................................................................. 105
$CPTDWay. Format ............................................................... 107
$CPTWDay. Format .............................................................. 108
$EBCDICw. Format ............................................................. 109
E8601DXw. Format ............................................................... 110
E8601LXw. Format ............................................................... 111
E8601TXw. Format ............................................................... 113
EUROW.d Format ............................................................... 114
EUROXw.d Format ............................................................. 117
HDATEw. Format ............................................................... 119
HEBDATEw. Format ........................................................... 120
$KANJIw. Format .............................................................. 121
$KANJIXw. Format ............................................................. 122
$LOGVSway. Format ........................................................... 123
$LOGVSRw. Format ............................................................ 124
MINGUOw. Format ............................................................. 126
NENGOw. Format .............................................................. 127
NLBESTw. Format .............................................................. 129
NLDATEw. Format ............................................................. 130
NLDATELw. Format ............................................................ 132
NLDATEMw. Format ........................................................... 132
NLDATEMDw. Format .......................................................... 133
NLDATEMDLw. Format ........................................................ 134
NLDATEMDMw. Format ......................................................... 135
NLDATEMDSw. Format ........................................................ 136
NLDATEMNw. Format ........................................................... 137
NLDATESw. Format ............................................................ 138
NLDATEWw. Format ........................................................... 139
NLDATEWNw. Format ........................................................... 140
NLDATEYMw. Format ........................................................... 141
NLDATEYMsw. Format ........................................................ 142
NLDATEYMMw. Format ........................................................ 143
NLDATEYMSw. Format ........................................................ 144
NLDATEYQw. Format ........................................................... 145
NLDATEYQlw. Format ........................................................... 146
<table>
<thead>
<tr>
<th>Format</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLDATYQMw. Format</td>
<td>147</td>
</tr>
<tr>
<td>NLDATYQSw. Format</td>
<td>148</td>
</tr>
<tr>
<td>NLDATYRw. Format</td>
<td>149</td>
</tr>
<tr>
<td>NLDATYWw. Format</td>
<td>150</td>
</tr>
<tr>
<td>NLDATMw. Format</td>
<td>151</td>
</tr>
<tr>
<td>NLDATMAPw. Format</td>
<td>152</td>
</tr>
<tr>
<td>NLDATMDTw. Format</td>
<td>153</td>
</tr>
<tr>
<td>NLDATMLw. Format</td>
<td>154</td>
</tr>
<tr>
<td>NLDATMMw. Format</td>
<td>155</td>
</tr>
<tr>
<td>NLDATMMDw. Format</td>
<td>155</td>
</tr>
<tr>
<td>NLDATMMDLw. Format</td>
<td>156</td>
</tr>
<tr>
<td>NLDATMMDMw. Format</td>
<td>157</td>
</tr>
<tr>
<td>NLDATMMDSw. Format</td>
<td>158</td>
</tr>
<tr>
<td>NLDATMMNw. Format</td>
<td>159</td>
</tr>
<tr>
<td>NLDATMSw. Format</td>
<td>160</td>
</tr>
<tr>
<td>NLDATMTMw. Format</td>
<td>161</td>
</tr>
<tr>
<td>NLDATMTZw. Format</td>
<td>162</td>
</tr>
<tr>
<td>NLDATMWw. Format</td>
<td>163</td>
</tr>
<tr>
<td>NLDATMWNw. Format</td>
<td>164</td>
</tr>
<tr>
<td>NLDATMWzw. Format</td>
<td>165</td>
</tr>
<tr>
<td>NLDATMYMw. Format</td>
<td>166</td>
</tr>
<tr>
<td>NLDATMYMLw. Format</td>
<td>167</td>
</tr>
<tr>
<td>NLDATMYMMMw. Format</td>
<td>168</td>
</tr>
<tr>
<td>NLDATMYMSw. Format</td>
<td>168</td>
</tr>
<tr>
<td>NLDATMYQw. Format</td>
<td>169</td>
</tr>
<tr>
<td>NLDATMYQLw. Format</td>
<td>170</td>
</tr>
<tr>
<td>NLDATMYQMw. Format</td>
<td>171</td>
</tr>
<tr>
<td>NLDATMYQSw. Format</td>
<td>172</td>
</tr>
<tr>
<td>NLDATMYRw. Format</td>
<td>173</td>
</tr>
<tr>
<td>NLDATMYWw. Format</td>
<td>174</td>
</tr>
<tr>
<td>NLDATMZw. Format</td>
<td>175</td>
</tr>
<tr>
<td>NLMNIAEDw.d Format</td>
<td>176</td>
</tr>
<tr>
<td>NLMNIAUDw.d Format</td>
<td>177</td>
</tr>
<tr>
<td>NLMNIBGnw.d Format</td>
<td>178</td>
</tr>
<tr>
<td>NLMNIBRLw.d Format</td>
<td>179</td>
</tr>
<tr>
<td>NLMNICAdw.d Format</td>
<td>180</td>
</tr>
<tr>
<td>NLMNICHFw.d Format</td>
<td>181</td>
</tr>
<tr>
<td>NLMNICNYw.d Format</td>
<td>182</td>
</tr>
<tr>
<td>NLMNICZKw.d Format</td>
<td>183</td>
</tr>
<tr>
<td>NLMNIDKKw.d Format</td>
<td>184</td>
</tr>
<tr>
<td>NLMNIEEkw.d Format</td>
<td>185</td>
</tr>
<tr>
<td>NLMNIEGPw.d Format</td>
<td>186</td>
</tr>
<tr>
<td>NLMNIEUrw.d Format</td>
<td>187</td>
</tr>
<tr>
<td>NLMNIGBPw.d Format</td>
<td>188</td>
</tr>
<tr>
<td>NLMNIHKDw.d Format</td>
<td>189</td>
</tr>
<tr>
<td>NLMNIHRKw.d Format</td>
<td>190</td>
</tr>
<tr>
<td>NLMNIHFw.d Format</td>
<td>191</td>
</tr>
<tr>
<td>NLMNIIDRw.d Format</td>
<td>192</td>
</tr>
<tr>
<td>NLMNIILSw.d Format</td>
<td>193</td>
</tr>
<tr>
<td>NLMNIINRw.d Format</td>
<td>194</td>
</tr>
<tr>
<td>NLMNIJJPYw.d Format</td>
<td>195</td>
</tr>
<tr>
<td>NLMNIKRWw.d Format</td>
<td>196</td>
</tr>
<tr>
<td>NLMNILTLw.d Format</td>
<td>197</td>
</tr>
<tr>
<td>NLMNILVLw.d Format</td>
<td>198</td>
</tr>
<tr>
<td>NLMNIMOPw.d Format</td>
<td>199</td>
</tr>
<tr>
<td>NLMNIMXNWw.d Format</td>
<td>200</td>
</tr>
</tbody>
</table>
Categories of NLS Formats

The following categories relate to NLS issues:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI Text Handling</td>
<td>Instructs SAS to write bidirectional data values from data variables.</td>
</tr>
<tr>
<td>Character</td>
<td>Instructs SAS to write character data values from character variables.</td>
</tr>
<tr>
<td>Currency Conversion</td>
<td>Instructs SAS to convert an amount from one currency to another currency.</td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DBCS</td>
<td>Instructs SAS to translate double-byte-character sets that are used in Asian languages.</td>
</tr>
<tr>
<td>Hebrew Text Handling</td>
<td>Instructs SAS to read Hebrew data from data variables.</td>
</tr>
<tr>
<td>International Date and Time</td>
<td>Instructs SAS to write data values from variables that represent dates, times, and datetimes.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Instructs SAS to write numeric data values from numeric variables.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI Text Handling</td>
<td>$BIDIw$ Format (p. 105)</td>
<td>Converts between a logically ordered string and a visually ordered string, by reversing the order of Hebrew and Arabic characters while preserving the order of Latin words and numbers.</td>
</tr>
<tr>
<td></td>
<td>$LOGVSw$ Format (p. 123)</td>
<td>Processes a character string that is in left-to-right-logical order, and then writes the character string in visual order.</td>
</tr>
<tr>
<td></td>
<td>$LOGVSRw$ Format (p. 124)</td>
<td>Processes a character string that is in right-to-left-logical order, and then writes the character string in visual order.</td>
</tr>
<tr>
<td></td>
<td>$VSLOGw$ Format (p. 290)</td>
<td>Processes a character string that is in visual order, and then writes the character string in left-to-right logical order.</td>
</tr>
<tr>
<td></td>
<td>$VSLOGRw$ Format (p. 291)</td>
<td>Processes a character string that is in visual order, and then writes the character string in right-to-left logical order.</td>
</tr>
<tr>
<td>Character</td>
<td>$EBCDICw$ Format (p. 109)</td>
<td>Converts native format character data to EBCDIC representation.</td>
</tr>
<tr>
<td></td>
<td>$UCS2Bw$ Format (p. 266)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS2BEw$ Format (p. 268)</td>
<td>Processes a character string that is in big-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UCS2Lw$ Format (p. 269)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS2LEw$ Format (p. 270)</td>
<td>Processes a character string that is in little-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$UCS2Xw$. Format (p. 271)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 16-bit, UCS2, Unicode encoding.</td>
<td></td>
</tr>
<tr>
<td>$UCS2XEw$. Format (p. 273)</td>
<td>Processes a character string that is in native-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4Bw$. Format (p. 274)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 32-bit, UCS4, Unicode encoding.</td>
<td></td>
</tr>
<tr>
<td>$UCS4BEw$. Format (p. 275)</td>
<td>Processes a character string that is in big-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4Lw$. Format (p. 276)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 32-bit, UCS4, Unicode encoding.</td>
<td></td>
</tr>
<tr>
<td>$UCS4LEw$. Format (p. 278)</td>
<td>Processes a character string that is in little-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UCS4Xw$. Format (p. 279)</td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 32-bit, UCS4, Unicode encoding.</td>
<td></td>
</tr>
<tr>
<td>$UCS4XEw$. Format (p. 280)</td>
<td>Processes a character string that is in native-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UESCw$. Format (p. 281)</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode escape (UESC) representation.</td>
<td></td>
</tr>
<tr>
<td>$UESCEw$. Format (p. 283)</td>
<td>Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UNCRw$. Format (p. 284)</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in numeric character representation (NCR).</td>
<td></td>
</tr>
<tr>
<td>$UNCREw$. Format (p. 285)</td>
<td>Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>$UPARENw$. Format (p. 286)</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation.</td>
<td></td>
</tr>
<tr>
<td>$UPARENEw$. Format (p. 287)</td>
<td>Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$UTF8Xw. Format (p. 288)</td>
<td></td>
<td>Processes a character string that is in the encoding of the current SAS session, and then writes the character string in universal transformation format (UTF-8) encoding.</td>
</tr>
<tr>
<td>$UTF8XEw. Format (p. 289)</td>
<td></td>
<td>Processes a character string that is in universal transformation format (UTF-8), and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td><strong>Date and Time</strong></td>
<td>B8601DXw. Format (p. 99)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone basic notation yyyymmdThhmmss+hhmm.</td>
</tr>
<tr>
<td></td>
<td>B8601LXw. Format (p. 101)</td>
<td>Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 basic notation yyyymmdThhmmss±hhmm.</td>
</tr>
<tr>
<td></td>
<td>B8601TXw. Format (p. 102)</td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 basic time notation hhmmss±hh:mm.</td>
</tr>
<tr>
<td></td>
<td>E8601DXw. Format (p. 110)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone extended notation yyyy-mm-ddThh:mm:ss±hh:mm.</td>
</tr>
<tr>
<td></td>
<td>E8601LXw. Format (p. 111)</td>
<td>Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 extended notation yyyy-mm-ddThh:mm:ss±hh:mm.</td>
</tr>
<tr>
<td></td>
<td>E8601TXw. Format (p. 113)</td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 extended time notation hh:mm:ss±hh:mm.</td>
</tr>
<tr>
<td></td>
<td>HDATEw. Format (p. 119)</td>
<td>Writes date values in the form yyyy mmmmm dd where dd is the day-of-the-month, mmmmm represents the month's name in Hebrew, and yyyy is the year.</td>
</tr>
<tr>
<td></td>
<td>HEBDATEw. Format (p. 120)</td>
<td>Writes date values according to the Jewish calendar.</td>
</tr>
<tr>
<td></td>
<td>MINGUOw. Format (p. 126)</td>
<td>Writes date values as Taiwanese dates in the form yyyyymmdd.</td>
</tr>
<tr>
<td></td>
<td>NENGOw. Format (p. 127)</td>
<td>Writes date values as Japanese dates in the form e.yymmdd.</td>
</tr>
<tr>
<td></td>
<td>NLDATEw. Format (p. 130)</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date.</td>
</tr>
<tr>
<td></td>
<td>NLDATELw. Format (p. 132)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date in the form, date, month, year.</td>
</tr>
<tr>
<td></td>
<td>NLDATEMw. Format (p. 132)</td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLDATEMDw. Format (p. 133)</td>
<td></td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the value as the name of the month and the day of the month.</td>
</tr>
<tr>
<td>NLDATEMDLw. Format (p. 134)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.</td>
</tr>
<tr>
<td>NLDATEMDMw. Format (p. 135)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.</td>
</tr>
<tr>
<td>NLDATEMDSw. Format (p. 136)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.</td>
</tr>
<tr>
<td>NLDATEMNW. Format (p. 137)</td>
<td></td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the value as the name of the month.</td>
</tr>
<tr>
<td>NLDATESw. Format (p. 138)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date string.</td>
</tr>
<tr>
<td>NLDATEWw. Format (p. 139)</td>
<td></td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the value as the date and the day of the week.</td>
</tr>
<tr>
<td>NLDATENWw. Format (p. 140)</td>
<td></td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the day of the week.</td>
</tr>
<tr>
<td>NLDATEMYw. Format (p. 141)</td>
<td></td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the name of the month.</td>
</tr>
<tr>
<td>NLDATEMYMLw. Format (p. 142)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the month and year.</td>
</tr>
<tr>
<td>NLDATEMYMMw. Format (p. 143)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date values as the month and year with abbreviations.</td>
</tr>
<tr>
<td>NLDATEMYMSw. Format (p. 144)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date and year.</td>
</tr>
<tr>
<td>NLDATEMYQw. Format (p. 145)</td>
<td></td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the quarter.</td>
</tr>
<tr>
<td>NLDATEMYQLw. Format (p. 146)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.</td>
</tr>
<tr>
<td>NLDATEMYQMw. Format (p. 147)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLDATEYQS&lt;sub&gt;W&lt;/sub&gt;. Format (p. 148)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (1–4) with numbers and delimiters.</td>
</tr>
<tr>
<td>NLDATEYR&lt;sub&gt;W&lt;/sub&gt;. Format (p. 149)</td>
<td></td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year.</td>
</tr>
<tr>
<td>NLDATEYW&lt;sub&gt;W&lt;/sub&gt;. Format (p. 150)</td>
<td></td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the week.</td>
</tr>
<tr>
<td>NLDATM&lt;sub&gt;W&lt;/sub&gt;. Format (p. 151)</td>
<td></td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.</td>
</tr>
<tr>
<td>NLDATMAP&lt;sub&gt;W&lt;/sub&gt;. Format (p. 152)</td>
<td></td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime with a.m. or p.m.</td>
</tr>
<tr>
<td>NLDATMDT&lt;sub&gt;W&lt;/sub&gt;. Format (p. 153)</td>
<td></td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month, day of the month and year.</td>
</tr>
<tr>
<td>NLDATML&lt;sub&gt;W&lt;/sub&gt;. Format (p. 154)</td>
<td></td>
<td>Converts a SAS datetime value to the date string of the specified locale, and then writes the date value as a date in the form, month, date, year, and time.</td>
</tr>
<tr>
<td>NLDATMM&lt;sub&gt;W&lt;/sub&gt;. Format (p. 155)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date and time with abbreviations for the month and time.</td>
</tr>
<tr>
<td>NLDATMMD&lt;sub&gt;W&lt;/sub&gt;. Format (p. 155)</td>
<td></td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month and the day of the month.</td>
</tr>
<tr>
<td>NLDATMMDL&lt;sub&gt;W&lt;/sub&gt;. Format (p. 156)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the full-length of the month and day of the month.</td>
</tr>
<tr>
<td>NLDATMMMD&lt;sub&gt;W&lt;/sub&gt;. Format (p. 157)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using abbreviations.</td>
</tr>
<tr>
<td>NLDATMMDS&lt;sub&gt;W&lt;/sub&gt;. Format (p. 158)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using numbers and delimiters.</td>
</tr>
<tr>
<td>NLDATMMN&lt;sub&gt;W&lt;/sub&gt;. Format (p. 159)</td>
<td></td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.</td>
</tr>
<tr>
<td>NLDATMS&lt;sub&gt;W&lt;/sub&gt;. Format (p. 160)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date in the form, MM/DD/YYYY.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLDATMTMw. Format (p. 161)</td>
<td></td>
<td>Converts the time portion of a SAS datetime value to the time-of-day value of the specified locale, and then writes the value as a time of day.</td>
</tr>
<tr>
<td>NLDATMTTZw. Format (p. 162)</td>
<td></td>
<td>Converts the time portion of the SAS datetime of the locale to the time of day and time zone.</td>
</tr>
<tr>
<td>NLDATMWWw. Format (p. 163)</td>
<td></td>
<td>Converts SAS datetime values to the locale sensitive datetime string as the day of the week and the datetime.</td>
</tr>
<tr>
<td>NLDATMWNw. Format (p. 164)</td>
<td></td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as the day of the week.</td>
</tr>
<tr>
<td>NLDATMWZw. Format (p. 165)</td>
<td></td>
<td>Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.</td>
</tr>
<tr>
<td>NLDATMYMw. Format (p. 166)</td>
<td></td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the month.</td>
</tr>
<tr>
<td>NLDATMYMLw. Format (p. 167)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.</td>
</tr>
<tr>
<td>NLDATMYMMw. Format (p. 168)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.</td>
</tr>
<tr>
<td>NLDATMYMSw. Format (p. 168)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the month and year with numbers and delimiters.</td>
</tr>
<tr>
<td>NLDATMYQw. Format (p. 169)</td>
<td></td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.</td>
</tr>
<tr>
<td>NLDATMYQLw. Format (p. 170)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter value (1–4) and the year.</td>
</tr>
<tr>
<td>NLDATMYQMw. Format (p. 171)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter (1–4) and then the year.</td>
</tr>
<tr>
<td>NLDATMYQSsw. Format (p. 172)</td>
<td></td>
<td>Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the quarter (1–4) using numbers and delimiters.</td>
</tr>
<tr>
<td>NLDATMYWRw. Format (p. 173)</td>
<td></td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.</td>
</tr>
<tr>
<td>NLDATMYWww. Format (p. 174)</td>
<td></td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>NLDATMZ</strong></td>
<td><strong>w. Format (p. 175)</strong></td>
<td>Converts SAS datetime values to the locale-sensitive datetime string as time zone and datetime.</td>
</tr>
<tr>
<td><strong>NLTIMAP</strong></td>
<td><strong>w. Format (p. 264)</strong></td>
<td>Converts a SAS time value to the time value of a specified locale, and then writes the value as a time value with a.m. or p.m. NLTIMAP also converts SAS date-time values.</td>
</tr>
<tr>
<td><strong>NLTIME</strong></td>
<td><strong>w. Format (p. 265)</strong></td>
<td>Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value. NLTIME also converts SAS date-time values.</td>
</tr>
<tr>
<td><strong>WEEKU</strong></td>
<td><strong>w. Format (p. 293)</strong></td>
<td>Writes a week number in decimal format by using the U algorithm.</td>
</tr>
<tr>
<td><strong>WEEKV</strong></td>
<td><strong>w. Format (p. 294)</strong></td>
<td>Writes a week number in decimal format by using the V algorithm.</td>
</tr>
<tr>
<td><strong>WEEKW</strong></td>
<td><strong>w. Format (p. 296)</strong></td>
<td>Writes a week number in decimal format by using the W algorithm.</td>
</tr>
<tr>
<td><strong>YYWEEKU</strong></td>
<td><strong>w. Format (p. 298)</strong></td>
<td>Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.</td>
</tr>
<tr>
<td><strong>YYWEEKV</strong></td>
<td><strong>w. Format (p. 300)</strong></td>
<td>Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.</td>
</tr>
<tr>
<td><strong>YYWEEKW</strong></td>
<td><strong>w. Format (p. 301)</strong></td>
<td>Writes a week number in decimal format by using the W algorithm, excluding the day-of-week information.</td>
</tr>
<tr>
<td><strong>DBCS</strong></td>
<td><strong>$KANJIw. Format (p. 121)</strong></td>
<td>Adds shift-code data to DBCS data.</td>
</tr>
<tr>
<td></td>
<td><strong>$KANJIXw. Format (p. 122)</strong></td>
<td>Removes shift-code data from DBCS data.</td>
</tr>
<tr>
<td><strong>Hebrew Text Handling</strong></td>
<td><strong>SCPTDWw. Format (p. 107)</strong></td>
<td>Processes a character string that is in Hebrew text, encoded in IBM-PC (cp862), and then writes the character string in Windows Hebrew encoding (cp 1255).</td>
</tr>
<tr>
<td></td>
<td><strong>SCPTWDw. Format (p. 108)</strong></td>
<td>Processes a character string that is encoded in Windows (cp1255), and then writes the character string in Hebrew DOS (cp862) encoding.</td>
</tr>
<tr>
<td><strong>ISO 8601</strong></td>
<td><strong>B8601DXw. Format (p. 99)</strong></td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone basic notation yyyymmdTthhmss+hhmm.</td>
</tr>
<tr>
<td></td>
<td><strong>B8601LXw. Format (p. 101)</strong></td>
<td>Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 basic notation yyyymmdTthhmss+</td>
</tr>
<tr>
<td></td>
<td><strong>B8601TXw. Format (p. 102)</strong></td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 basic time notation hhmmss+</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>E8601DXw. Format (p. 110)</td>
<td>Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone extended notation yyyy-mm-ddThh:mm:ss+hh:mm.</td>
</tr>
<tr>
<td></td>
<td>E8601LXw. Format (p. 111)</td>
<td>Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 extended notation yyyy-mm-ddThh:mm:ss+</td>
</tr>
<tr>
<td></td>
<td>E8601TXw. Format (p. 113)</td>
<td>Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 extended time notation hh:mm:ss+</td>
</tr>
<tr>
<td>Numeric</td>
<td>BESTDOTXw. Format (p. 104)</td>
<td>Specifies that SAS choose the best notation and use a dot as a decimal separator.</td>
</tr>
<tr>
<td></td>
<td>EUROw.d Format (p. 114)</td>
<td>Writes numeric values with a leading euro symbol (E), a comma that separates every three digits, and a period that separates the decimal fraction.</td>
</tr>
<tr>
<td></td>
<td>EUROXw.d Format (p. 117)</td>
<td>Writes numeric values with a leading euro symbol (E), a period that separates every three digits, and a comma that separates the decimal fraction.</td>
</tr>
<tr>
<td></td>
<td>NLBESTw. Format (p. 129)</td>
<td>Writes the best numerical notation based on the locale.</td>
</tr>
<tr>
<td></td>
<td>NLMNIAEDw.d Format (p. 176)</td>
<td>Writes the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>NLMNIAUDw.d Format (p. 177)</td>
<td>Writes the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIBGNw.d Format (p. 178)</td>
<td>Writes the monetary format of the international expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>NLMNIBRLw.d Format (p. 179)</td>
<td>Writes the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNICADw.d Format (p. 180)</td>
<td>Writes the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNICCHFw.d Format (p. 181)</td>
<td>Writes the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNICNYw.d Format (p. 182)</td>
<td>Writes the monetary format of the international expression for China.</td>
</tr>
<tr>
<td></td>
<td>NLMNICZKw.d Format (p. 183)</td>
<td>Writes the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>NLMNIDKKw.d Format (p. 184)</td>
<td>Writes the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLMNIEEKw.d Format (p. 185)</td>
<td></td>
<td>Writes the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td>NLMNIEGPw.d Format (p. 186)</td>
<td></td>
<td>Writes the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td>NLMNIEURw.d Format (p. 187)</td>
<td></td>
<td>Writes the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td>NLMNIGBPw.d Format (p. 188)</td>
<td></td>
<td>Writes the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td>NLMNIHKDw.d Format (p. 189)</td>
<td></td>
<td>Writes the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td>NLMNIHRKw.d Format (p. 190)</td>
<td></td>
<td>Writes the monetary format of the international expression for Croatia.</td>
</tr>
<tr>
<td>NLMNIHUFw.d Format (p. 191)</td>
<td></td>
<td>Writes the monetary format of the international expression for Hungary.</td>
</tr>
<tr>
<td>NLMNIIDRW.d Format (p. 192)</td>
<td></td>
<td>Writes the monetary format of the international expression for Indonesia.</td>
</tr>
<tr>
<td>NLMNIILSW.d Format (p. 193)</td>
<td></td>
<td>Writes the monetary format of the international expression for Israel.</td>
</tr>
<tr>
<td>NLMNIINRW.d Format (p. 194)</td>
<td></td>
<td>Writes the monetary format of the international expression for India.</td>
</tr>
<tr>
<td>NLMNIJPYw.d Format (p. 195)</td>
<td></td>
<td>Writes the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td>NLMNIKRWW.d Format (p. 196)</td>
<td></td>
<td>Writes the monetary format of the international expression for South Korea.</td>
</tr>
<tr>
<td>NLMNILTLW.d Format (p. 197)</td>
<td></td>
<td>Writes the monetary format of the international expression for Lithuania.</td>
</tr>
<tr>
<td>NLMNILVWLW.d Format (p. 198)</td>
<td></td>
<td>Writes the monetary format of the international expression for Latvia.</td>
</tr>
<tr>
<td>NLMNIMPW.d Format (p. 199)</td>
<td></td>
<td>Writes the monetary format of the international expression for Macau.</td>
</tr>
<tr>
<td>NLMNIMGXNW.d Format (p. 200)</td>
<td></td>
<td>Writes the monetary format of the international expression for Mexico.</td>
</tr>
<tr>
<td>NLMNIMYRW.d Format (p. 201)</td>
<td></td>
<td>Writes the monetary format of the international expression for Malaysia.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLMNINOKw.d Format (p. 202)</td>
<td>Writes the monetary format of the international expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>NLMININZDw.d Format (p. 203)</td>
<td>Writes the monetary format of the international expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>NLMNIPLNw.d Format (p. 204)</td>
<td>Writes the monetary format of the international expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>NLMNIRUBw.d Format (p. 205)</td>
<td>Writes the monetary format of the international expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>NLMNISEKw.d Format (p. 206)</td>
<td>Writes the monetary format of the international expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>NLMNISGDw.d Format (p. 207)</td>
<td>Writes the monetary format of the international expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>NLMNITHBw.d Format (p. 208)</td>
<td>Writes the monetary format of the international expression for Thailand.</td>
</tr>
<tr>
<td></td>
<td>NLMNITRYw.d Format (p. 209)</td>
<td>Writes the monetary format of the international expression for Turkey.</td>
</tr>
<tr>
<td></td>
<td>NLMNITWDw.d Format (p. 210)</td>
<td>Writes the monetary format of the international expression for Taiwan.</td>
</tr>
<tr>
<td></td>
<td>NLMNIUSDw.d Format (p. 211)</td>
<td>Writes the monetary format of the international expression for Puerto Rico and the United States.</td>
</tr>
<tr>
<td></td>
<td>NLMNIZARw.d Format (p. 212)</td>
<td>Writes the monetary format of the international expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>NLMNLAEDx.d Format (p. 213)</td>
<td>Writes the monetary format of the local expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>NLMNLAUDw.d Format (p. 214)</td>
<td>Writes the monetary format of the local expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBGNw.d Format (p. 215)</td>
<td>Writes the monetary format of the local expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>NLMNLBRLw.d Format (p. 216)</td>
<td>Writes the monetary format of the local expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCADw.d Format (p. 217)</td>
<td>Writes the monetary format of the local expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCHEw.d Format (p. 218)</td>
<td>Writes the monetary format of the local expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNLCNYw.d Format (p. 219)</td>
<td>Writes the monetary format of the local expression for China.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLMNLNZKw.d Format (p. 220)</td>
<td>Writes the monetary format of the local expression for the Czech Republic.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLDKkw.d Format (p. 221)</td>
<td>Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLEEKw.d Format (p. 222)</td>
<td>Writes the monetary format of the local expression for Estonia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLEG_PW.d Format (p. 223)</td>
<td>Writes the monetary format of the local expression for Egypt.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLEURw.d Format (p. 224)</td>
<td>Writes the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLGBPw.d Format (p. 225)</td>
<td>Writes the monetary format of the local expression for the United Kingdom.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLHKDw.d Format (p. 226)</td>
<td>Writes the monetary format of the local expression for Hong Kong.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLHRKw.d Format (p. 227)</td>
<td>Writes the monetary format of the local expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLHUFw.d Format (p. 228)</td>
<td>Writes the monetary format of the local expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLIDRW.d Format (p. 229)</td>
<td>Writes the monetary format of the local expression for Indonesia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLILSw.d Format (p. 230)</td>
<td>Writes the monetary format of the local expression for Israel.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLINRW.d Format (p. 231)</td>
<td>Writes the monetary format of the local expression for India.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNJPYW.d Format (p. 232)</td>
<td>Writes the monetary format of the international expression for Japan.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNKRWW.d Format (p. 233)</td>
<td>Writes the monetary format of the local expression for South Korea.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLLTLW.d Format (p. 234)</td>
<td>Writes the monetary format of the local expression for Lithuania.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLLVW.d Format (p. 235)</td>
<td>Writes the monetary format of the local expression for Latvia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNLMOPw.d Format (p. 236)</td>
<td>Writes the monetary format of the local expression for Macau.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLMNLXMNw.d Format (p. 237)</td>
<td>Writes the monetary format of the local expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>NLMNLMYRw.d Format (p. 238)</td>
<td>Writes the monetary format of the local expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNOKw.d Format (p. 239)</td>
<td>Writes the monetary format of the local expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNZDw.d Format (p. 240)</td>
<td>Writes the monetary format of the local expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>NLMNLPLNW.d Format (p. 241)</td>
<td>Writes the monetary format of the local expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLRUBw.d Format (p. 242)</td>
<td>Writes the monetary format of the local expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLSEKw.d Format (p. 243)</td>
<td>Writes the monetary format of the local expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>NLMNLSGDW.d Format (p. 244)</td>
<td>Writes the monetary format of the local expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTTHBW.d Format (p. 245)</td>
<td>Writes the monetary format of the local expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTRYw.d Format (p. 246)</td>
<td>Writes the monetary format of the local expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>NLMNLW.DW.d Format (p. 247)</td>
<td>Writes the monetary format of the local expression for Taiwan.</td>
<td></td>
</tr>
<tr>
<td>NLMNLUSDw.d Format (p. 248)</td>
<td>Writes the monetary format of the local expression for Puerto Rico and the United States.</td>
<td></td>
</tr>
<tr>
<td>NLMNLZARw.d Format (p. 249)</td>
<td>Writes the monetary format of the local expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>NLMNYw.d Format (p. 250)</td>
<td>Writes the monetary format of the local expression in the specified locale using local currency.</td>
<td></td>
</tr>
<tr>
<td>NLMNYiw.d Format (p. 251)</td>
<td>Writes the monetary format of the international expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLMNUMw.d Format (p. 252)</td>
<td>Writes the numeric format of the local expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLMNMIw.d Format (p. 254)</td>
<td>Writes the numeric format of the international expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>NLPCTw.d Format (p. 255)</td>
<td>Writes percentage data of the local expression in the specified locale.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLPCTIw.d Format (p. 256)</td>
<td></td>
<td>Writes percentage data of the international expression in the specified locale.</td>
</tr>
<tr>
<td>NLPCTNw.d Format (p. 258)</td>
<td></td>
<td>Produces percentages, using a minus sign for negative values.</td>
</tr>
<tr>
<td>NLPCTPw.d Format (p. 259)</td>
<td></td>
<td>Writes locale-specific numeric values as percentages.</td>
</tr>
<tr>
<td>NLPVALUEw.d Format (p. 260)</td>
<td></td>
<td>Writes p-values of the local expression in the specified locale.</td>
</tr>
<tr>
<td>NLSTRMONw.d Format (p. 261)</td>
<td></td>
<td>Writes the month name in the specified locale.</td>
</tr>
<tr>
<td>NLSTRTWQw.d Format (p. 262)</td>
<td></td>
<td>Writes a numeric value as the quarter-of-the-year in the specified locale.</td>
</tr>
<tr>
<td>NLSTRWKw.d Format (p. 263)</td>
<td></td>
<td>Writes a numeric value as the day-of-the-week in the specified locale.</td>
</tr>
<tr>
<td>YENw.d Format (p. 297)</td>
<td></td>
<td>Writes numeric values with yen signs, commas, and decimal points.</td>
</tr>
</tbody>
</table>

### Dictionary

**B8601DXw. Format**

Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone basic notation `yyyyymmddThhmmss +hhmm`.

- **Categories:** Date and Time
- **Alignment:** Left
- **Supports:** ISO 8601 Element 5.4.1, complete representation

#### Syntax

*B8601DXw*.

#### Syntax Description

*<w>* specifies the width of the output field.

**Default** 26
Details

UTC values specify a date and a time that are based on the zero meridian in Greenwich, England. Using this format, SAS converts a datetime value to the UTC value and determines the user local date and time by using the value of the TIMEZONE= system option. If the TIMEZONE= option is not set, then the user local date and time are based on the local date and time. The B8601DX format writes SAS datetime values by using the following ISO 8601 basic datetime notation:

- \textit{yyyyymmddThhmmss+hhmm}

\textit{yyyy}

is a four-digit year.

\textit{mm}

is a two-digit month (zero padded) between 01 and 12.

\textit{dd}

is a two-digit day of the month (zero padded) between 01 and 31.

\textit{hh}

is a two-digit hour (zero padded) between 00 and 23.

\textit{mm}

is a two-digit minute (zero padded) between 00 and 59.

\textit{ss}

is a two-digit second (zero padded) between 00 and 59.

\textit{+\text{–}hhmm}

is an hour and minute signed offset from zero meridian time. The offset must be \textit{+\text{–}hhmm} (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +0200 indicates a two-hour time difference to the east of the zero meridian, and –0600 indicates a six-hour time difference to the west of the zero meridian.

Restriction: The shorter form \textit{+\text{–}hh} is not supported.

Example

The first example uses the local time to determine the time and the time zone offset. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em> ;</td>
<td>20130201T073456-0500</td>
</tr>
<tr>
<td>t='01Feb2013T12:34:56'dt ;</td>
<td></td>
</tr>
<tr>
<td>put t b8601dx.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
### Statement

```r
options timezone='America/Adak';
data _null_;  
t='01Feb2013T12:34:56'dt ;  
put t b8601dx.;  
run;
```

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options timezone='America/Adak';</code></td>
<td></td>
</tr>
<tr>
<td><code>data _null_;</code></td>
<td></td>
</tr>
<tr>
<td><code>t='01Feb2013T12:34:56'dt ;</code></td>
<td></td>
</tr>
<tr>
<td><code>put t b8601dx.;</code></td>
<td></td>
</tr>
<tr>
<td><code>run;</code></td>
<td></td>
</tr>
</tbody>
</table>

| 20130201T023456-1000 |

### See Also

“Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations” in *SAS Formats and Informats: Reference*

---

**B8601LXw. Format**

Writes datetime values as local time by appending a time zone offset difference between the local time and UTC, using the ISO 8601 basic notation `yyyyymmddThhmmss+|–hhmm`.

- **Categories:** Date and Time
- **ISO 8601**
- **Alignment:** Right
- **Supports:** ISO 8601 Elements 5.3.3 and 5.3.4.2

### Syntax

**B8601LXw**

**Syntax Description**

- `w` specifies the width of the output field.

- **Default** 26
- **Range** 20–35

### Details

The B8601LX format writes datetime values without making any adjustments, and appends the UTC time zone offset for the local SAS session by using the ISO 8601 basic datetime notation:

- `yyyyymmddThhmmss+|–hhmm`

- `yyyy` is a four-digit year.

- `mm` is a two-digit month (zero padded) between 01 and 12.

- `dd` is a two-digit day of the month (zero padded) between 01 and 31.
hh
is a two-digit hour (zero padded) between 00 and 23.

mm
is a two-digit minute (zero padded) between 00 and 59.

ss
is a two-digit second (zero padded) between 00 and 59.

+|–hhmm
is an hour and minute signed offset from zero meridian time. The offset must be +|–
hhmm (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of
the zero meridian. For example, +0200 indicates a two-hour time difference to the east
of the zero meridian, and –0600 indicates a six-hour time difference to the west of
the zero meridian.

Restriction: The shorter form +|–hh is not supported.

Example
This PUT statement writes the time for the Eastern Standard time zone:

blx='01Feb2013T12:34:56'dt;
put blx b8601lx.;

<table>
<thead>
<tr>
<th>Value of blx</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1675341296</td>
<td>20130201T123456-0500</td>
</tr>
</tbody>
</table>

See Also
“Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations”
in *SAS Formats and Informats: Reference*

**B8601TXw. Format**

Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by
using the ISO 8601 basic time notation hhmmss+|−hhmm.

**Categories:** Date and Time
ISO 8601

**Alignment:** Right

**Supports:** ISO 8601 Elements 5.3.3 and 5.3.4

**Syntax**

B8601TXw.
Syntax Description

\( w \)

specifies the width of the output field.

Default 14

Range 9–20

Details

UTC values specify a time based on the zero meridian in Greenwich, England. Using this format, SAS converts a time value to the UTC value and determines the user local time by using the TIMEZONE= system option. If the TIMEZONE= option is not set, then the user local time is based on the local time. The B8601TX format writes SAS datetime values by using the following ISO 8601 basic time notation:

- \( hhmmss \pm hhmm \)

- \( hh \) is a two-digit hour (zero padded) between 00 and 23.

- \( mm \) is a two-digit minute (zero padded) between 00 and 59.

- \( ss \) is a two-digit second (zero padded) between 00 and 59.

- \( \pm hhmm \) is an hour and minute signed offset from zero meridian time. The offset must be \( \pm hhmm \) (that is, \( \pm \) and four characters).

Use \( + \) for time zones east of the zero meridian, and use \( - \) for time zones west of the zero meridian. For example, \( +0200 \) indicates a two-hour time difference to the east of the zero meridian, and \( -0600 \) indicates a six-hour time difference to the west of the zero meridian.

Restriction: The shorter form \( \pm hh \) is not supported.

When SAS reads a UTC time by using the B8601TZ informat, and the adjusted time is greater than 24 hours or less than 00 hours, SAS adjusts the value so that the time is between 000000 and 240000. If the B8601TX format attempts to format a time outside of this time range, the time is formatted with asterisks to indicate that the value is out of range.

Example

The first example uses the local time to determine the time and the time zone offset. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>t='12:34:56't;</td>
<td>073456-0500</td>
</tr>
<tr>
<td>put t b8601tx.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
options timezone='America/Adak';
data _null_;  
t='12:34:56't;  
put t b8601tx.;  
run;

See Also

“Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations” in SAS Formats and Informats: Reference

BESTDOTXw. Format

Specifies that SAS choose the best notation and use a dot as a decimal separator.

Category: Numeric
Alignment: Right

Syntax

BESTDOTXw.

Syntax Description

w

specifies the width of the output field.

Default 12
Range 1–32

Tip If you print numbers between 0 and .01 exclusively, use a field width of at least 7 to avoid excessive rounding. If you print numbers between 0 and -01 exclusively, use a field width of at least 8.

Details

If the NLDECSEPARATOR system option is disabled, the BESTw and BESTDOTXw formats process data the same way. If the NLDECSEPARATOR system option is enabled, then the results from the BEST and BESTDOTX formats are different. See the following table to understand the differences:

<table>
<thead>
<tr>
<th>LOCALE option</th>
<th>Default decimal separator character for the locale</th>
<th>NLDECSEPARATOR option</th>
<th>Separator character used by BESTw.</th>
<th>Separator character used by BESTDOTXw.</th>
</tr>
</thead>
<tbody>
<tr>
<td>en_US</td>
<td>Dot</td>
<td>Disabled (default)</td>
<td>Dot</td>
<td>Dot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enabled</td>
<td>Dot</td>
<td>Dot</td>
</tr>
<tr>
<td>LOCALE option</td>
<td>Default decimal separator character for the locale</td>
<td>NLDECSEPARATOR option</td>
<td>Separator character used by BESTw.</td>
<td>Separator character used by BESTDOTXw.</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------</td>
<td>------------------------</td>
<td>-------------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>fr_FR</td>
<td>Comma</td>
<td>Disabled (default)</td>
<td>Dot</td>
<td>Dot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enabled</td>
<td>Comma</td>
<td>Dot</td>
</tr>
</tbody>
</table>

### Example

The following code produces results using the BESTDOTX format with the NLDECSEPARATOR option enabled and the locale set to fr_FR.

#### Statements

```sas
options nld locale=fr_FR;
data _null_;  
x=1.2;     
  call symput('macx', put(x, BESTDOTX.));  
  put x; /* Result is printed as "1,2" */

y=put(x, BESTDOTX.); /* Result is printed as "1.2" */
run;

&put &macx; /* Result is printed as "1.2" */
run;

data _null_;  
  bx=&macx; /* Succeeds because the value of &macx is 1.2 */
  put bx;  
  run;
```

#### Results

<table>
<thead>
<tr>
<th>Statements</th>
<th>1.2</th>
<th>1.2</th>
<th>1.2</th>
<th>1.2</th>
</tr>
</thead>
</table>

### See Also

- “BESTw. Format” in SAS Formats and Informats: Reference
- “NLDECSEPARATOR System Option” on page 598

### $BIDIw. Format

Converts between a logically ordered string and a visually ordered string, by reversing the order of Hebrew and Arabic characters while preserving the order of Latin words and numbers.

**Category:** BIDI Text Handling  
**Alignment:** Left

### Syntax

$BIDIw.$
Syntax Description

\( w \)

specifies the width of the output field.

Default

1 if \( w \) is not specified

Range

1–32767

Details

In the Windows operating environment, Hebrew and Arabic text is stored in logical order. The text is stored in the order in which it is written and not necessarily as it is displayed. However, in other operating environments, Hebrew text is stored in the same order it is displayed. SAS users can encounter Hebrew and Arabic text that is reversed. Such situations can occur when you use SAS/CONNECT or other software to transfer SAS data sets or reports with Hebrew and Arabic text from a visual operating environment to a logical one. The $BIDI format is a format that reverses Hebrew and Arabic text while maintaining the order of numbers and Latin-1 words.

Operating Environment Information

In mainframe operating environments, this format is designed to work with NewCode Hebrew and Arabic. Some mainframe operating environments might experience unsatisfactory results, because they use the OldCode Hebrew or Arabic encoding. There is a hotfix for this encoding on: SAS Institute’s Web site.

Comparisons

The $BIDIw. format performs a reversing function similar to the $REVERJw. format, which writes character data in reverse order and preserves blanks. $BIDIw. behaves in the following way:

- $BIDIw. reverses the order of words and numbers in a specified string, preserving blanks. Latin-1 words and numbers themselves are not reversed, only their order in the string.
- When $BIDI encounters a word consisting of Hebrew or Arabic characters in the text string, the characters in the Hebrew or Arabic word are reversed. The position of the Hebrew or Arabic word is reversed in the string.

Example

This example demonstrates how $BIDIw. reverses Hebrew characters. The Hebrew is reversed in the string. The Hebrew characters in the words are also reversed.

```sas
data;
  a='שלום עבד ותא נ abc 123';
  b1 = put (a,$bidi20.);
  put b=;
  b2 = put (b,$bidi20.);
  put b=;
run;
```

The following lines are written to the SAS log:

```
b1=123 abc נ abc ותא 123
b2=שלום עבד ותא נ abc 123
```
$CPTDWw. Format

Processes a character string that is in Hebrew text, encoded in IBM-PC (cp862), and then writes the character string in Windows Hebrew encoding (cp 1255).

Category: Hebrew Text Handling
Alignment: Left

Syntax

$CPTDWw.

Syntax Description

w

specifies the width of the output field.

Default 200
Range 1–32767

Comparisons

The $CPTDWw. format performs processing that is the opposite of the $CPTWDw. format.

Example

The following example uses the input value of “808182x.”

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $cptdw3.;</td>
<td>118</td>
</tr>
</tbody>
</table>

See Also

Format:
- “$CPTWDw. Format” on page 108

Informs:
- “$CPTDWw. Informat” on page 435
- “$CPTWDw. Informat” on page 436
$\texttt{CPTWD}w. \textit{Format}

Processes a character string that is encoded in Windows (cp1255), and then writes the character string in Hebrew DOS (cp862) encoding.

\begin{itemize}
  \item \textbf{Category:} Hebrew Text Handling
  \item \textbf{Alignment:} Left
\end{itemize}

\textbf{Syntax}

$\texttt{CPTWD}w.$

\textbf{Syntax Description}

\textit{w}

Specifies the width of the output field.

\begin{tabular}{ll}
  \textbf{Default} & 200 \\
  \textbf{Range} & 1–32767
\end{tabular}

\textbf{Comparisons}

The $\texttt{CPTWD}w.$ format performs processing that is the opposite of the $\texttt{CPTDW}w.$ format.

\textbf{Example}

The following example uses the input value of “א För”.

\begin{tabular}{ll}
  \textbf{Statement} & \textbf{Result} \\\n  \hline
  \texttt{put text $\text{cptwd3.};} & ٍ٦٠, \\
\end{tabular}

\textbf{See Also}

\textbf{Format:}

- “$\texttt{CPTDW}w.$ Format” on page 107

\textbf{Informats:}

- “$\texttt{CPTDW}w.$ Informat” on page 435
- “$\texttt{CPTWD}w.$ Informat” on page 436
$EBCDICw. Format

Converts native format character data to EBCDIC representation.

**Category:** Character

**Alignment:** Left

**Syntax**

$EBCDICw.$

**Syntax Description**

*w* specifies the width of the output field.

- **Default:** 1
- **Range:** 1–32767

**Details**

If EBCDIC is the native format, no conversion occurs.

On ASCII systems, the $EBCDICw.$ format is based on the default encoding value of the LOCALE= option that is specified when SAS starts. For example, if the locale was set to en_US locale, the default encoding that is used by the $EBCDICw.$ format is Open ed-1047. If the locale is de_DE (German_Germany), the default encoding that is used by the $EBCDICw.$ format is Open ed–1141. For a list of locales and encoding values, see Table 20.2 on page 704.

You can specify the translation table that is used to map characters between EBCDIC and ASCII by using the MAPEBCDIC2ASCII system option. For more information, see “MAPEBCDIC2ASCII= System Option” on page 597.

**Comparisons**

- On ASCII systems, $EBCDICw.$ converts ASCII character data to EBCDIC.
- On all other systems, $EBCDICw.$ behaves like the $CHARw.$ format.

**Example**

put name $ebcdic3.;

<table>
<thead>
<tr>
<th>Value of name</th>
<th>Locale Value Is Fr_US</th>
<th>Locale Value Is fr_FR</th>
<th>Locale Value Is de_DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>C1C2C3</td>
<td>C1C2C3</td>
<td>C1C2C3</td>
</tr>
<tr>
<td>{â}</td>
<td>C043D0</td>
<td>514354</td>
<td>43C0DC</td>
</tr>
<tr>
<td>[©]</td>
<td>AD7CBD</td>
<td>9044B5</td>
<td>63B5FC</td>
</tr>
</tbody>
</table>
The results are shown as hexadecimal representations of EBCDIC codes for characters. Each two hexadecimal characters correspond to one byte of binary data, and each byte corresponds to one character.

**E8601DXw. Format**

Adjusts a Coordinated Universal Time (UTC) datetime value to the user local date and time. Then, writes the local date and time by using the ISO 8601 datetime and time zone extended notation `yyyy-mm-ddThh:mm:ss+hh:mm`.

**Syntax**

`E8601DXw`.  

**Syntax Description**

`w` specifies the width of the output field.  

**Default** 26  

**Range** 20–35

**Details**

UTC values specify a date and time that are based on the zero meridian in Greenwich, England. Using this format, SAS converts a datetime value to the UTC value and determines the user local date and time by using the value of the `TIMEZONE=` system option. If the `TIMEZONE=` option is not set, the user local date and time are based on the local date and time. The E8601DX format writes SAS datetime values by using the following ISO 8601 basic datetime notation:

- `yyyy-mm-ddThh:mm:ss+hh:mm`

  `yyyy` is a four-digit year.  

  `mm` is a two-digit month (zero padded) between 01 and 12.  

  `dd` is a two-digit day of the month (zero padded) between 01 and 31.  

  `hh` is a two-digit hour (zero padded) between 00 and 23.  

  `mm` is a two-digit minute (zero padded) between 00 and 59.  

  `ss` is a two-digit second (zero padded) between 00 and 59.
$\pm hh:mm$ is an hour and minute signed offset from zero meridian time. The offset must be $\pm hh:mm$ (that is, + or – and four characters).

Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +02:00 indicates a two-hour time difference to the east of the zero meridian, and –06:00 indicates a six-hour time difference to the west of the zero meridian.

**Restriction:** The shorter form $\pm hh$ is not supported.

**Example**

The first example uses the local time to determine the time. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; t='01Feb2013T12:34:56'dt; put t e8601dx.; run;</td>
<td>2013-02-01T07:34:56-05:00</td>
</tr>
<tr>
<td>options timezone='America/Adak'; data <em>null</em>; t='01Feb2013T12:34:56'dt; put t e8601dx.; run;</td>
<td>2013-02-01T02:34:56-10:00</td>
</tr>
</tbody>
</table>

**See Also**

“Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations” in *SAS Formats and Informats: Reference*
Details

The E8601LX format writes datetime values without making any adjustments, and appends the UTC time zone offset for the local SAS session by using the ISO 8601 basic datetime notation:

- \( yyyy-mm-ddThh:mm:ss+hh:mm \)

  \( yyyy \)
  
  is a four-digit year.

  \( mm \)
  
  is a two-digit month (zero padded) between 01 and 12.

  \( dd \)
  
  is a two-digit day of the month (zero padded) between 01 and 31.

  \( hh \)
  
  is a two-digit hour (zero padded) between 00 and 23.

  \( mm \)
  
  is a two-digit minute (zero padded) between 00 and 59.

  \( ss \)
  
  is a two-digit second (zero padded) between 00 and 59.

  \(+|−hh:mm\)

  is an hour and minute signed offset from zero meridian time. The offset must be \(+|−hh:mm\) (that is, + or − and four characters).

  Use + for time zones east of the zero meridian, and use − for time zones west of the zero meridian. For example, +02:00 indicates a two-hour time difference to the east of the zero meridian, and −06:00 indicates a six-hour time difference to the west of the zero meridian.

  **Restriction:** The shorter form \(+|−hh\) is not supported.

Example

This PUT statement writes the time for the Eastern time zone:

```
blx='01Feb2013T12:34:56'dt;
p put blx e8601lx.;
```

<table>
<thead>
<tr>
<th>Value of blx</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1675341296</td>
<td>2013-02-01T12:34:56-05:00</td>
</tr>
</tbody>
</table>

See Also

“Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations” in *SAS Formats and Informats: Reference*
E8601TXw. Format

Adjusts a Coordinated Universal Time (UTC) value to the user local time. Then, writes the local time by using the ISO 8601 extended time notation \( hh:mm:ss+|–hh:mm \).

**Categories:** Date and Time
ISO 8601

**Alignment:** Right

**Supports:** ISO 8601 Elements 5.3.3 and 5.3.4

**Syntax**

\( \text{E8601TXw} \)

**Syntax Description**

\( w \)

specifies the width of the output field.

**Default** 14

**Range** 9–20

**Details**

UTC values specify a time based on the zero meridian in Greenwich, England. Using this format, SAS converts a time value to the UTC value and determines the user local time by using the value of the TIMEZONE= system option. If the TIMEZONE= option is not set, the user local time is based on the local time. The E8601TX format writes SAS datetime values by using the following ISO 8601 basic time notation:

- \( hh:mm:ss+|–hh:mm \)

  \( hh \)

  is a two-digit hour (zero padded) between 00 and 23.

  \( mm \)

  is a two-digit minute (zero padded) between 00 and 59.

  \( ss \)

  is a two-digit second (zero padded) between 00 and 59.

  \( +|–hh:mm \)

  is an hour and minute signed offset from zero meridian time. The offset must be \(+|–hh:mm \) (that is, + or – and four characters).

  Use + for time zones east of the zero meridian, and use – for time zones west of the zero meridian. For example, +0200 indicates a two-hour time difference to the east of the zero meridian, and –0600 indicates a six-hour time difference to the west of the zero meridian.

  **Restriction:** The shorter form \(+|–hh \) is not supported.

When SAS reads a UTC time by using the E8601TZ informat, and the adjusted time is greater than 24 hours or less than 00 hours, SAS adjusts the value so that the time is between 000000 and 240000. If the E8601TX format attempts to format a time outside
of this time range, the time is formatted with asterisks to indicate that the value is out of range.

Example
The first example uses the local time to determine the time and the time zone offset. The second example changes the time zone to America/Adak, which is Hawaii-Aleutian Time.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
</table>
| data _null_;
t='12:34:56't;
put t e8601tx.;
run;         | 07:34:56-05:00 |
| options timezone='America/Adak';
data _null_;
t='12:34:56't;
put t e8601tx.;
run;         | 02:34:56-10:00 |

See Also
“Working with Dates and Times By Using the ISO 8601 Basic and Extended Notations” in SAS Formats and Informats: Reference

EUROw.d Format
Writes numeric values with a leading euro symbol (E), a comma that separates every three digits, and a period that separates the decimal fraction.

Category: Numeric
Alignment: Right

Syntax
EUROw.d

Syntax Description

\[ w \]
- specifies the width of the output field.

Default 6
Range 1-32
Tip If you want the euro symbol to be part of the output, be sure to choose an adequate width.

\[ d \]
- specifies the number of digits to the right of the decimal point in the numeric value.
Comparisons

- The EURO\textit{w.d} format is similar to the EUROX\textit{w.d} format, but EUROX\textit{w.d} format reverses the roles of the decimal point and the comma. This convention is common in European countries.

- The EURO\textit{w.d} format is similar to the DOLLAR\textit{w.d} format, except that DOLLAR\textit{w.d} format writes a leading dollar sign instead of the euro symbol.

Example

These examples use 1254.71 as the value of amount.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put amount euro10.2;</td>
<td>E1,254.71</td>
</tr>
<tr>
<td>put amount euro5.;</td>
<td>1,255</td>
</tr>
<tr>
<td>put amount euro9.2;</td>
<td>E1,254.71</td>
</tr>
<tr>
<td>put amount euro15.3;</td>
<td>E1,254.710</td>
</tr>
</tbody>
</table>

```sas
data _null_;  
  input x;  
  put x euro10.2;  
  put x euro5.;  
  put x euro9.2;  
  put x euro15.3;  
  datalines;  
  1254.71  
;  
run;  
SAS Log:  
 E1,254.71  
1,255  
E1,254.71  
E1,254.710  

/* This code determines the default length. */  
data _null_;  
  input x;  
  put x euro.;  
  datalines;  
  1  
  22
```
/* This code determines the range. */
data _null_;  
input x;  
put x euro5.;  
put x euro6.;  
put x euro7.;  
put x euro8.;  
put x euro9.;  
put x euro9.2;  
put x euro10.;  
put x euro10.2;  
put x euro10.4;  
put x euro11.;  
put x euro11.3;  
put x euro12.;  
put x euro12.2;  
put x euro13.;  
put x euro13.2;  
datalines;
333  
4444  
55555  
666666  
7777777  
88888888  
999999999  
1234561234  
;run;
SAS Log:  
datalines;  
E1  
E22  
E333  
E4,444  
55,555  
666666  
7.78E6  
8.89E7  
1E9  
1.23E9  
NOTE: At least one W.D format was too small for the number to be printed.  
The decimal may be shifted by the "BEST" format.
EUROXw.d Format

Writes numeric values with a leading euro symbol (€), a period that separates every three digits, and a comma that separates the decimal fraction.

**Category:** Numeric

**Alignment:** Right

**Syntax**

`EUROXw.d`

**Syntax Description**

`w`

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>1-32</td>
</tr>
<tr>
<td>Tip</td>
<td>If you want the euro symbol to be part of the output, be sure to choose an adequate width.</td>
</tr>
</tbody>
</table>

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

<table>
<thead>
<tr>
<th>Default</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>0-31</td>
</tr>
<tr>
<td>Requirement</td>
<td>must be less than <code>w</code></td>
</tr>
</tbody>
</table>

**Comparisons**

- The `EUROXw.d` format is similar to the `EUROw.d` format, but `EUROw.d` format reverses the roles of the comma and the decimal point. This convention is common in English-speaking countries.
- The `EUROXw.d` format is similar to the `DOLLARXw.d` format, except that `DOLLARXw.d` format writes a leading dollar sign instead of the euro symbol.

**Example**

These examples use 1254.71 as the value of amount.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put amount eurox10.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox5.;</td>
<td>1.255</td>
</tr>
<tr>
<td>put amount eurox9.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox15.3;</td>
<td>E1.254,710</td>
</tr>
</tbody>
</table>

```
data _null_;  
  input x;  
  put x eurox10.2;  
  put x eurox5.;  
  put x eurox9.2;  
  put x eurox15.3;  
  datalines;  
1254.71  
; run;  
SAS Log:  
E1.254,71  
1.255  
E1.254,71  
   E1.254,710  
/* This code determines the default length. */  
data _null_;  
  input x;  
  put x eurox.;  
  datalines;  
1  
22  
333  
4444  
55555  
666666  
7777777  
88888888  
999999999  
1234561234  
;run;  
SAS Log:  
E1  
E22  
E333  
E4.444  
S5.555  
666666  
7.78E6  
8.89E7  
1B9  
1.23B9
```
Note: At least one W.D format was too small for the number to be printed. The decimal might be shifted by the "BEST" format.

See Also

Format:

- “EUROw.d Format” on page 114

Informats:

- “EUROw.d Informat” on page 437
- “EUROXw.d Informat” on page 439

HDATExw. Format

Writers date values in the form yyyy mmmmm dd where dd is the day-of-the-month, mmmmm represents the month's name in Hebrew, and yyyy is the year.

Category: Date and Time
Alignment: Right

Syntax

HDATExw.

Syntax Description

w

specifies the width of the output field.

Note: Use widths 9, 11, 15, or 17 for the best view.

Default 17
Range 9–17

Details

The HDATExw. format writes the SAS date value in the form yyyy mmmmm dd:

yyyy
is the year

mmmmm
is English month name written in Hebrew letters

dd
is the day-of-the-month

Example

The following example uses the input value of 15780, which is the SAS date of March 16, 2003.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day hdate9.;</td>
<td>03  י&quot;ע 16</td>
</tr>
<tr>
<td>put day hdate11.;</td>
<td>2003  י&quot;ע 16</td>
</tr>
<tr>
<td>put day hdate17.;</td>
<td>2003  י&quot;ע 16</td>
</tr>
</tbody>
</table>

See Also

Format:
- “HEBDATEw. Format” on page 120

**HEBDATEw. Format**

Writes date values according to the Jewish calendar.

- **Category:** Date and Time
- **Alignment:** Right

**Syntax**

HEBDATEw.

**Syntax Description**

w

specifies the width of the output field.

- **Default:** 16
- **Range:** 7–24
- **Tip:** When using a non-Hebrew encoding, asterisks appear instead of the Hebrew letters.

**Details**

The Jewish calendar is a combined solar and lunar calendar. Years are counted from the creation of the world, which according to Jewish history, occurred 3760 years and three months before the commencement of the Christian. You must add 3761, beginning in the autumn of a specified year in the Gregorian calendar to calculate the Hebrew year.

The HEBDATEw. format writes the SAS date value according to the Jewish calendar. The date is written in one of the following formats:

long

📅 י"ע  המסהי

ר'

📅 י"ע  המסהי

📅 י"ע  המסהי
The following example uses the input value of 15780, which is the SAS date of March 16, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day hebdate13.;</td>
<td>י&quot;ב/12/תשס&quot;ב</td>
</tr>
<tr>
<td>put day hebdate16.;</td>
<td>רו' ב' אדר תשס&quot;ב</td>
</tr>
<tr>
<td>put day hebdate24.;</td>
<td>לאז'ו י' אדר-ב' ה'תשס&quot;כ</td>
</tr>
</tbody>
</table>

See Also

Informat:
• “HDATEw. Format” on page 119

$KANJlw. Format

Adds shift-code data to DBCS data.

Category: DBCS
Alignment: Left

Syntax

$KANJlw:

Syntax Description

w

specifies the width of the output field.

Range

The minimum width of the format is 2 + (length of shift code used on the current DBCSTYPE= setting)*2
Restriction  The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.

Details

The $KANJI format adds shift-code data to DBCS data that does not have shift-code data. If the input data is blank, shift-code data is not added.

The $KANJI format processes host-mainframe data, but $KANJI can be used on other platforms. If you use the $KANJI format on non-EBDIC (non-modal encoding) hosts, the data does not change.

See Also

Formats:
- “$KANJIXw. Format” on page 122

Informat:
- “$KANJIw. Informat” on page 440
- “$KANJIXw. Informat” on page 441

System Option:
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 583

$KANJIXw. Format

Removes shift-code data from DBCS data.

Category:  DBCS
Alignment:  Left

Syntax

$KANJIXw

Syntax Description

w

specifies the width of the output field.

Range  The minimum width of the format is 2.

Restriction  The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.
Details

The $KANJIX format removes shift-code data from DBCS data. The input data length must be \(2 + (SO/SI \text{ length}) \times 2\). The data must start with SO and end with SI, unless single-byte data is returned.

The $KANJIX format processes host mainframe data, but $KANJIX can be used on other platforms. If you use the $KANJIX format on non-EBCDIC (non-modal encoding) hosts, the data does not change.

See Also

Format:

- “$KANJIIw. Format” on page 121

Informs:

- “$KANJIw. Informat” on page 440
- “$KANJIXw. Informat” on page 441

System Option:

- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 583

$LOGVSw. Format

Processes a character string that is in left-to-right-logical order, and then writes the character string in visual order.

**Category:** BIDI Text Handling

**Alignment:** Left

**Syntax**

$LOGVSw.

**Syntax Description**

\(w\)

specifies the width of the output field.

**Default**  200

**Range**  1–32767

Details

The $LOGVSw. format is used when you store logical-ordered text on a visual server.  

**Note:** If the $LOGVSw. format is not accessible, then the Hebrew or Arabic portion of the data is reversed.
Comparisons

The $LOGVSw. format performs processing that is the opposite of the $LOGVSRw. format.

Example

The following example uses the Hebrew input value of “flight”.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12.;</td>
<td>ﬂIGHT</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “computer”.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12.;</td>
<td>ﻃﺎﺑـ ﻃ ﺟ computer</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$LOGVSRw. Format” on page 124

Informats:
- “$LOGVSRw. Informat” on page 443
- “$LOGVSw. Informat” on page 442

$LOGVSRw. Format

Processes a character string that is in right-to-left-logical order, and then writes the character string in visual order.

Category: BIDI Text Handling
Alignment: Left

Syntax

$LOGVSRw.
**Syntax Description**

\[ w \]

- specifies the width of the output field.
- **Default**: 200
- **Range**: 1–32767

**Details**

The $LOGVSRw$ format is used when you store logical-ordered text on a visual server. The Hebrew or Arabic portion of the text is reversed if the $LOGVS$ format is not on the server.

**Comparisons**

The $LOGVSRw$ format performs processing that is opposite of the $LOGVS$ format.

**Example**

The following example uses the Hebrew input value of "_flight".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvsr12.;</td>
<td>flight ע&quot;י &quot;</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "_computer".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvsr12.;</td>
<td>ذا &quot;computer&quot;</td>
</tr>
</tbody>
</table>

**See Also**

- “$LOGVSw Format” on page 123

**Formats:**

- “$LOGVSw Format” on page 123

**Informats:**

- “$LOGVSw Informat” on page 442
- “$LOGVSRw Informat” on page 443
MINGUOW Format

Writes date values as Taiwanese dates in the form yyyymmdd.

**Category:** Date and Time

**Alignment:** Left

---

**Syntax**

MINGUOW:

**Syntax Description**

w

specifies the width of the output field.

**Default** 8

**Range** 1–10

---

**Details**

The MINGUOW format writes SAS date values in the form yyyymmdd, where

- **yyy** is an integer that represents the year.
- **mm** is an integer that represents the month.
- **dd** is an integer that represents the day of the month.

The Taiwanese calendar uses 1912 as the base year (01/01/01 is January 1, 1912). Dates before 1912 appear as a series of asterisks. Year values do not roll around after 100 years. Instead, they continue to increase.

---

**Example**

The example table uses the following input values:

- 12054 is the SAS date value that corresponds to January 1, 1993.
- 18993 is the SAS date value that corresponds to January 1, 2012.
- -20088 is the SAS date value that corresponds to January 1, 1905.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put(12054,minguo7.); put x=;</td>
<td>x=820101</td>
</tr>
<tr>
<td>x=put(12054,minguo9.); put x=;</td>
<td>x=82/01/01</td>
</tr>
</tbody>
</table>
NENGOw. Format

Writes date values as Japanese dates in the form e.yymmdd.

**Category:** Date and Time

**Alignment:** Left

### Syntax

NENGOw

### Syntax Description

w

specifies the width of the output field.

- **Default:** 10
- **Range:** 2–10
Details

The NENGOw. format writes SAS date values in the form $e.yymmdd$, where

$e$

is the first letter of the name of the emperor (Meiji, Taisho, Showa, or Heisei).

$yy$

is an integer that represents the year.

$mm$

is an integer that represents the month.

$dd$

is an integer that represents the day of the month.

If the width is too small, SAS omits the period.

Example

The example table uses the input value of 15342, which is the SAS date value that corresponds to January 2, 2002.

```sas
data _null_;    
  date=15342;    
  put date nengo3.;    
  put date nengo6.;    
  put date nengo8.;    
  put date nengo9.;    
  put date nengo10.;    
run
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date nengo3.;</td>
<td>H14</td>
</tr>
<tr>
<td>put date nengo6.;</td>
<td>H14/01</td>
</tr>
<tr>
<td>put date nengo8.;</td>
<td>H.140102</td>
</tr>
<tr>
<td>put date nengo9.;</td>
<td>H14/01/02</td>
</tr>
<tr>
<td>put date nengo10.;</td>
<td>H.14/01/02</td>
</tr>
</tbody>
</table>

See Also

Informat:

• “NENGOw. Informat” on page 446
NLBESTw. Format

 Writes the best numerical notation based on the locale.

**Category:** Numeric  
**Alignment:** Right

### Syntax

**NLBESTw:**

### Syntax Description

**w**  
specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>1–32</td>
</tr>
</tbody>
</table>

**Tip**  
If you print numbers between 0 and .01 exclusively, then use a field width of at least 7 to avoid excessive rounding. If you print numbers between 0 and -.01 exclusively, use a field width of at least 8.

### Details

The NLBEST format writes the best numerical value based on the locale's decimal point and the sign mark's location. NLBEST is similar to the BEST format. For more information, see the BEST format in the *SAS Formats and Informats: Reference*.

### Example

The following code produces results based on the locale:

```sas
x=-1257000  
  put x nlbest6.;  
  put x nlbest3.;  
  put "=====";  
  x=-0.1  
  put x nlbest6.;  
  put x nlbest3.;  
  put "=====";  
  x=0.1  
  put x nlbest6.;  
  put x nlbest3.;  
  put "=====";  
  x=1257000  
  put x nlbest6.;  
  put x nlbest3.;
```
### Locales

<table>
<thead>
<tr>
<th>Locales</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>locale=English_UnitedStates</td>
<td>-126E4</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>-.1</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>1.26E6</td>
</tr>
<tr>
<td></td>
<td>1E6</td>
</tr>
<tr>
<td>locale=German_Germany</td>
<td>-126E4</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>-.1</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>1,26E6</td>
</tr>
<tr>
<td></td>
<td>1E6</td>
</tr>
<tr>
<td>locale=ar_BH</td>
<td>126E4-</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>0.1-</td>
</tr>
<tr>
<td></td>
<td>.1-</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>1,26E6</td>
</tr>
<tr>
<td></td>
<td>1E6</td>
</tr>
</tbody>
</table>

---

**NLDATEw. Format**

Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date.
Category: Date and Time
Alignment: Left

Syntax
NLDATEw.

Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 20
Range 10–200

Comparisons
NLDATEw. is similar to DATEw. and WORDDATEw. except that NLDATEw. is locale-specific.

Example
These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; put day nldate.;</td>
<td>February 24, 2003</td>
</tr>
<tr>
<td>options locale=German_Germany; put day nldate.;</td>
<td>24. Februar 2003</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “NLDATEMNw. Format” on page 137
- “NLDATEWw. Format” on page 139
- “NLDATEWNw. Format” on page 140
NLDATELw. Format
Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date in the form, date, month, year.

**Category:** Date and Time

**Alignment:** Left

### Syntax
NLDATELw.

### Syntax Description

- **w**
  - Specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default:** 18

**Range:** 2–220

### Details
NLDATEL outputs the date in a long-uniform pattern with the full length of the month and week names.

### Example
This example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+---- NLDATEL min=2 default=18 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 19, 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19/2012</td>
</tr>
<tr>
<td>put &quot;---- NLDATEL min=2 default=18 max=200 ----&quot;;</td>
<td>Nov 19, 2012</td>
</tr>
<tr>
<td>put dy nldatel.;</td>
<td>November 19, 2012</td>
</tr>
<tr>
<td>put dy nldateln10.;</td>
<td>November 19, 2012</td>
</tr>
<tr>
<td>put dy nldateln12.;</td>
<td>November 19, 2012</td>
</tr>
<tr>
<td>put dy nldateln18.;</td>
<td>November 19, 2012</td>
</tr>
<tr>
<td>put dy nldateln200.;</td>
<td>November 19, 2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATEMw. Format
Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date.
Category: Date and Time
Alignment: Left

Syntax
NLDATEM

Syntax Description

\( w \)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 10
Range 2–200

Details

NLDATEM outputs the date in a medium-uniform pattern such as Nov 19, 2012.

Example

This example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATEM min=2 default=14 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Nov 19, 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19/12</td>
</tr>
<tr>
<td>put &quot;----- NLDATEM min=2 default=14 max=200 ----&quot;;</td>
<td>Nov 19, 2012</td>
</tr>
<tr>
<td>put dy nldatem.;</td>
<td>Nov 19, 2012</td>
</tr>
<tr>
<td>put dy nldatem8.;</td>
<td></td>
</tr>
<tr>
<td>put dy nldatem14.;</td>
<td></td>
</tr>
<tr>
<td>put dy nldatem200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATEGw. Format

Converts the SAS date value to the date value of the specified locale, and then writes the value as the name of the month and the day of the month.

Category: Date and Time
Alignment: Left

Syntax

NLDATEGw.
Syntax Description

\(w\)

specifies the width of the output field.

Default 16

Range 6-200

Example

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put 1 nldatemd.;</td>
<td>January 02</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLDATEYMw. Format” on page 141

NLDATEMDLw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.

Category: Date and Time

Alignment: Left

Syntax

NLDATEMDLw.

Syntax Description

\(w\)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 12

Range 5–200

Details

NLDATEMDL outputs the date in a long-uniform pattern with full length of the month and the day, such as November 19.
## Example

The following example uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
</tr>
<tr>
<td>dt = datetime();</td>
</tr>
<tr>
<td>dy = date();</td>
</tr>
<tr>
<td>put &quot;+--- NLDATEMDL min=5 default=12 max=200 ---+&quot;;</td>
</tr>
<tr>
<td>put dy nldatemdl.;</td>
</tr>
<tr>
<td>put dy nldatemdl5.;</td>
</tr>
<tr>
<td>put dy nldatemdl9.;</td>
</tr>
<tr>
<td>put dy nldatemdl12.;</td>
</tr>
<tr>
<td>put dy nldatemdl200.;</td>
</tr>
<tr>
<td>run;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 19</td>
</tr>
<tr>
<td>11/19</td>
</tr>
<tr>
<td>Nov 19</td>
</tr>
<tr>
<td>November 19</td>
</tr>
<tr>
<td>November 19</td>
</tr>
</tbody>
</table>

---

### NLDATEMDMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.

- **Category:** Date and Time
- **Alignment:** Left

### Syntax

**NLDATEMDMw.**

### Syntax Description

<table>
<thead>
<tr>
<th>w</th>
</tr>
</thead>
</table>

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 9
- **Range:** 5–200

### Details

NLDATEMDM outputs the date in a medium-uniform pattern with abbreviation of the month and the day using numbers and delimiters, such as Nov 19.

### Example

The following example uses the date of November 19, 2012.
**NLDatemdsw Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

`NLDatemdsw.`

#### Syntax Description

`w`

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 5  
- **Range:** 5–200

### Details

NLDatemd outputs the date in a short-uniform pattern in full length of the month and the day using numbers and delimiters, such as MM/dd.

### Example

The following example uses the date of November 19, 2012.

```sas
data _null_
  dt = datetime()
  dy = date()
  put "---- NLDatemd min=5 default=9 max=200 ----";
  put dy nldatemd.;
  put dy nldatemd5.;
  put dy nldatemd9.;
  put dy nldatemd200.;
run;
```

The output is:

```
Nov 19 11/19 Nov 19 Nov 19
```

---
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATEMDS min=5 default=5 max=200 ---+";  
put dy nldatemds.;  
put dy nldatemds5.;  
put dy nldatemds5.;  
put dy nldatemds200.;  
run;

---+--- NLDATEMDS min=5 default=5 max=200 ---+
   11/19
   11/19
   11/19
   11/19

---+--- NLDATEMDS min=5 default=5 max=200 ---+

NLDATEMNW. Format

Converts a SAS date value to the date value of the specified locale, and then writes the value as the name of the month.

Category: Date and Time
Alignment: Left

Syntax

NLDATEMNW.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the name of the month to fit the format width.

Default 9
Range 4–200

Comparisons

NLDATEMNW. is similar to MONNAMEw. except that NLDATEMNW. is locale-specific.

Example

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

Statements | Results
---+---+---|
----+----1

-----+-----1
### NLDATESw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date string.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

#### Syntax

NLDATESw.

#### Syntax Description

- **w**
  - specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.
  - Default: 10
  - Range: 2–200

#### Details

NLDATES outputs the date in a short-uniform pattern that contains only numbers and delimiters, such as mm/dd/yyyy.

#### Example

This example uses the date November 19, 2012.
### Statements

```sas
data _null_;  
dt = datetime();  
dy = date();  
put +++- NLDATES min=2 default=10 max=200 ++++;  
put dy nldates.;  
put dy nldates8.;  
put dy nldates10.;  
put dy nldates200.;  
run;
```

### Results

```
+--- NLDATES min=2 default=10 max=200 ---+
11/19/2012
11/19/2012
11/19/2012
11/19/2012
```

---

### NLDATEWw. Format

Converts a SAS date value to the date value of the specified locale, and then writes the value as the date and the day of the week.

- **Category:** Date and Time
- **Alignment:** Left

#### Syntax

```
NLDATEWw.
```

#### Syntax Description

- `w` specifies the width of the output field. If necessary, SAS abbreviates the date and the day of the week to fit the format width.

- **Default:** 29
- **Range:** 10–200

#### Details

The NLDATEW format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 79 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

#### Comparisons

NLDATEWw. is similar to WEEKDATEw. except that NLDATEWw. is locale specific.

#### Example

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.
Statements | Results
---|---

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; date=15760; x=put(date,nldate.); y=put(date,nldate20.); z=put(date,nldate200.); run;</td>
<td>Monday, February 24, 2003 Mon, Feb 24, 2003 Monday, February 24, 2003</td>
</tr>
<tr>
<td>options locale=German_Germany; date=15760; x=put(date,nldate.); y=put(date,nldate20.); z=put(date,nldate200.); run;</td>
<td>Mo., 24. Feb 2003 Mo., 24. Feb 2003 Montag, 24. Februar 2003</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “NLDATEw. Format” on page 130
- “NLDATEWNw. Format” on page 137
- “NLDATEWNw. Format” on page 140

NLDATEWNw. Format

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the day of the week.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLDATEWNw.

Syntax Description

`w`

specifies the width of the output field. If necessary, SAS abbreviates the day of the week to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>4–200</td>
</tr>
</tbody>
</table>
Comparisons

NLDATEWNw. is similar to DOWNAMEw. except that NLDATEWNw. is locale-specific.

Example

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>Monday</td>
</tr>
<tr>
<td>put date nldatewn.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>Montag</td>
</tr>
<tr>
<td>put date nldatewn.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “NLDATEw. Format” on page 130
- “NLDATEMNw. Format” on page 137
- “NLDATEWw. Format” on page 139

NLDATEYMw. Format

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the name of the month.

Category: Date and Time
Alignment: Left

Syntax

NLDATEYMw.

Syntax Description

w

specifies the width of the output field.

Default 16
Range 6–200
Details

If you specify a width of 6, but your data is larger than 6, your output contains asterisks: 
d=******. To remove the asterisks, you can use PROC LOCALEDATA. The following example uses PROC LOCALEDATA to output the date without the asterisks:

PROC LOCALEDATA;
  LOAD SASLOCALE;
  MODIFY key=DATE_YYMM_SHORT_FORMAT value='%b %y' ;
  SAVE REGISTRY / _ALL_ syntax=SAS;

data _null_;  
format d nldateym6.;  
d = '17OCT14'd;  
put d=;  
run;  
   d=Oct 14

Example

This example uses the spanish_Spain locale option, and the date of August 2010.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=spanish_Spain;</td>
<td></td>
</tr>
<tr>
<td>data_null_;</td>
<td>agosto de 2010</td>
</tr>
<tr>
<td>dy=today();</td>
<td>ago de 10</td>
</tr>
<tr>
<td>x=put(dy, nldateym.);</td>
<td>agosto de 2010</td>
</tr>
<tr>
<td>y=put(dy, nldateym12.);</td>
<td></td>
</tr>
<tr>
<td>z=put(dy, nldateym200.);</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLDATEMDw. Format” on page 133

NLDATEYMLw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the month and year.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATEYMLw.
Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 14

Range 5–200

Details

NLDATEYML outputs the date in a long-uniform pattern with abbreviations for the month and year, such as November 2012.

Example

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATEYML min=5 default=14 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/2012</td>
</tr>
<tr>
<td>put &quot;+---- NLDATEYML min=5 default=14 max=200 ----&quot;;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dy nldateyml.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml7.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml11.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml14.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dy nldateyml200.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATEYMMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date values as the month and year with abbreviations.

Category: Date and Time
Alignment: Left

Syntax

NLDATEYMMw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 11
Details

NLDATEYMS outputs the date in a medium-uniform pattern with abbreviations for the month and year, such as Nov 2012.

Example

The following example uses the date November 19, 2012.

```
Statements

data _null_;
  dt = datetime();
  dy = date();
  put "--- NLDATEYMM min=5 default=11 max=200 ---";
  put dy nldateymm.;
  put dy nldateymm7.;
  put dy nldateymm11.;
  put dy nldateymm200.;
run;

--- NLDATEYMM min=5 default=11 max=200 ---
Nov 2012
11/2012
Nov 2012
Nov 2012
```

NLDATEYMSw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date and year.

Category: Date and Time
Alignment: Left

Syntax

NLDATEYMSw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 7
Range 5–200

Details

NLDATEYMS outputs the date in a short-uniform pattern with numbers and delimiters such as mm/yyy.
Example

This example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATEYMS min=5 default=7 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>11/2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/12</td>
</tr>
<tr>
<td>put &quot;+--- NLDATEYMS min=5 default=7 max=200 ---+&quot;;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dy nldateyms.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dy nldateyms5.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dy nldateyms7.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dy nldateyms200.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLDATEYQw. Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the quarter.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

NLDATEYQw.

**Syntax Description**

w  

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4–200</td>
</tr>
</tbody>
</table>

**Details**

The NLDATEYQ format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 79 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

**Example**

This example uses the fr_FR locale option.
NLDATEYQLw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year's quarter value (Q1–Q4) using abbreviations.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

NLDATEYQLw.

### Syntax Description

**w**  
specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>4–200</td>
</tr>
</tbody>
</table>

### Details

NLDATEYQL outputs the date in a long-uniform pattern with full length for the year and year’s quarter value, such as 4th quarter 2012.

### Example

The following example uses the date November 19, 2012.
NLDATEYQMW. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (Q1–Q4) using abbreviations.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

NLDATEYQMW.

**Syntax Description**

\( w \)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default**

7

**Range**

4–200

**Details**

NLDATEYQM outputs the date in a medium-uniform pattern with abbreviations for the year and year’s quarter value, such as Q4 2012.

**Example**

The following example uses the date November 19, 2012.
```sas
data _null_;
  dt = datetime();
  dy = date();
  put "+--- NLDATEYQM min=4 default=7 max=200 ---+
      Q4 2012
      2012/4
      Q4 2012
      Q4 2012
      Q4 2012
      Q4 2012
      Q4 2012
    run;
```

### NLDATEYQS Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the year’s quarter value (1–4) with numbers and delimiters.

**Category:** Date and Time  
**Alignment:** Left

#### Syntax

`NLDATEYQS w.

#### Syntax Description

`w`

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 6
- **Range:** 4–200

#### Details

NLDATEYQS outputs the date in a short-uniform pattern with numbers and delimiters for the year and year’s quarter value, such as 2012/4.

#### Example

The following example uses the date November 19, 2012.
Statements |
---|---|
data _null_; |
dt = datetime(); |
dy = date(); |
put "+---- NLDATEYQS min=4 default=6 max=200 ----+
2012/4 |
12/4 |
2012/4 |
2012/4 |
run; |

Results |
---|---|

**NLDATEYRw. Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year.

- **Category:** Date and Time
- **Alignment:** Left

**Syntax**

NLDATEYRw.

**Syntax Description**

- `w`
  - specifies the width of the output field.
  - **Default:** 16
  - **Range:** 2–200

**Example**

This example uses the fr_FR locale option.
Statements

options locale=fr_FR;
data _null_
;
dy=today();
dt=datetime();
put "+--- NLDATEYR min=2 default=16 max=200 ---+
";
put dy nldateyr.;
put dy nldateyr2.;
put dy nldateyr8.;
put dy nldateyr200.;
run;

Results

+--- NLDATEYR min=2 default=16 max=200 ---+
2008
08
2008
2008

NLDATEYWw. Format

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the week.

Category: Date and Time
Alignment: Left

Syntax

NLDATEWYWw.

Syntax Description

w

specifies the width of the output field.

Default 16
Range 5–200

Example

This example uses the fr_FR locale option.
### Statements

```sas
options locale=fr_FR;
data _null_

dy=today();
dt=datetime();
put "+--- NLDATEYW min=5 default=16 max=200 ---+";
put ' 16' +5 dy nldateyw.;
put ' 5' +5 dy nldateyw5.;
put ' 8' +5 dy nldateyw8.;
put ' 32' +5 dy nldateyw32.;
put '200' +5 dy nldateyw200.;
run;
```

### Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16 Week 33 2008</td>
<td>5 *****</td>
</tr>
<tr>
<td>32 Week 33 2008</td>
<td>200</td>
</tr>
<tr>
<td>Week 33 2008</td>
<td></td>
</tr>
</tbody>
</table>

---

## NLDATMw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

`NLDATMw.`

### Syntax Description

- **w**
  
  Specifies the width of the output field. If necessary, SAS abbreviates the datetime value to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10–200</td>
</tr>
</tbody>
</table>

###Comparisons

The NLDATMw. format is similar to the DATETIMEw. format except that the NLDATMw. format is locale-specific.

### Example

These examples use the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003.
### See Also

**Formats:**

- “NLDATMAPw. Format” on page 152
- “NLDATMTMw. Format” on page 161
- “NLDATMWw. Format” on page 163

---

#### NLDATMAPw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime with a.m. or p.m.

- **Category:** Date and Time
- **Alignment:** Left

#### Syntax

NLDATMAPw.

#### Syntax Description

- `w` specifies the width of the output field. If necessary, SAS abbreviates the date-time value to fit the format width.

  - **Default** 32
  - **Range** 16–200

#### Comparisons

The NLDATMAPw. format is similar to DATEAMPMw. except that the NLDATMAPw. format is locale-specific.

#### Example

These examples use the input value of 1361709583, which is the SAS date-time value that corresponds to 12:39:43 p.m. on February 24, 2003.
NLDATMDTw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month, day of the month and year.

Category: Date and Time
Alignment: Left

Syntax

NLDATMDTw.

Syntax Description

w

specifies the width of the output field

Default 18

Range 10-200

Example

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; put event nldatmap.;</td>
<td>February 24, 2003 12:39:43 PM</td>
</tr>
<tr>
<td>options locale=Spanish_Mexico; put event nldatmap.;</td>
<td>24/02/2003 12:39:43 PM</td>
</tr>
</tbody>
</table>

See Also

Formats:

• “NLDATMw. Format” on page 151
• “NLDATMTMw. Format” on page 161
• “NLDATMWw. Format” on page 163
See Also

Formats:
- “NLDATMMDw. Format” on page 155

**NLDATMLw. Format**

Converts a SAS datetime value to the date string of the specified locale, and then writes the date value as a date in the form, month, date, year, and time.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

NLDATMLw.

**Syntax Description**

w  

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default** 30  
**Range** 9–200

**Details**

NLDATML outputs the date in a long-uniform pattern with the full length of the month date, year, and time, such as, November 19, 2012 02:57:44 PM.

**Example**

This example uses the date November 19, 2012.
### Statements

```sas
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATML min=9 default=30 max=200 ---+
       November 19, 2012 02:57:44 PM
       11/19/2012 14:57:44
      Nov 19, 2012 02:57:44 PM
      November 19, 2012 02:57:44 PM

       November 19, 2012 02:57:44 PM
       November 19, 2012 02:57:44 PM
       November 19, 2012 02:57:44 PM

run;
```

### Results

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>+--- NLDATML min=9 default=30 max=200 ---+</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>11/19/2012 14:57:44</td>
<td>Nov 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>Nov 19, 2012 02:57:44 PM</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>November 19, 2012 02:57:44 PM</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
<tr>
<td>November 19, 2012 02:57:44 PM</td>
<td>November 19, 2012 02:57:44 PM</td>
</tr>
</tbody>
</table>

---

### NLDATMMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date and time with abbreviations for the month and time.

- **Category:** Date and Time
- **Alignment:** Left

#### Syntax

```
NLDATMMw.
```

#### Syntax Description

`w`

- Specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.
  - **Default:** 24
  - **Range:** 9–200

#### Details

NLDATMM outputs the date in a medium-uniform pattern with abbreviations of the month and week names, such as Nov 19, 2012 02:51:40 PM.

---

### NLDATMMDw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month and the day of the month.

- **Category:** Date and Time
- **Alignment:** Left
**Syntax**

\[ \text{NLDATMMD}_w. \]

**Syntax Description**

\[ w \]

specifies the width of the output field.

Default 16

Range 6–200

**Example**

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=en_US;</td>
<td>January 02</td>
</tr>
<tr>
<td>data <em>null</em></td>
<td></td>
</tr>
<tr>
<td>x=put(86400,nldatmmd.);</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “NLDATMYMw. Format” on page 166

---

**NLDATMMDLw. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the full-length of the month and day of the month.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

\[ \text{NLDATMMDL}_w. \]

**Syntax Description**

\[ w \]

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 9
Details

NLDATMMDL outputs the date in a long-uniform pattern with full-length of the month and the day, such as November 19.

Example

The following example uses the date of November 19, 2012.

```
data _null_;  
dt = datetime();  
dy = date();  
put "+--- NLDATMMDL min=5 default=12 max=200 ---+";  
put dt nldatmmdl.;  
put dt nldatmmdl5.;  
put dt nldatmmdl9.;  
put dt nldatmmdl12.;  
put dt nldatmmdl200.;  
run;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt = datetime();</td>
<td>November 19</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMMDL min=5 default=12 max=200 ---+&quot;;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmmdl.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl5.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl9.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl12.;</td>
<td>November 19</td>
</tr>
<tr>
<td>put dt nldatmmdl200.;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMMDMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using abbreviations.

- **Category:** Date and Time
- **Alignment:** Left

Syntax

NLDATMMDMw.

Syntax Description

w  

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 9
- **Range:** 5–200

Details

NLDATMMDM outputs the date in a medium-uniform pattern with abbreviations of the month and the day, such as Nov 19.
### Example

The following example uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMMDD min=5 default=9 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Nov 19</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/19</td>
</tr>
<tr>
<td>put &quot;+---- NLDATMMDD min=5 default=9 max=200 ----+&quot;;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dt nldatmmmdm.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dt nldatmmdd5.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dt nldatmmdd9.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>put dt nldatmmdd200.;</td>
<td>Nov 19</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

---

### NLDATMMDDsw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and day of the month using numbers and delimiters.

- **Category:** Date and Time
- **Alignment:** Left

### Syntax

NLDATMMDDsw.

### Syntax Description

\( w \)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 5
- **Range:** 5–200

### Details

NLDATMMDD outputs the date in a short-uniform pattern with numbers and delimiters of the month and the day, such as 11/19.

### Example

The following example uses the date of November 19, 2012.
Statements | Results
---|---
data _null_; | +++- NLDATMMDS min=5 default=5 max=200 ----+
dt = datetime(); | 11/19
dy = date(); | 11/19
put "+++ NLDATMMDS min=5 default=5 max=200 +++"; | 11/19
put dt nldatmmds.; | 11/19
put dt nldatmmds5.; | 11/19
put dt nldatmmds5.; | 11/19
put dt nldatmmds200.; | 11/19
run;

**NLDATMMNw. Format**

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.

- **Category:** Date and Time
- **Alignment:** Left

### Syntax

**NLDATMMNw.**

### Syntax Description

**w**

- specifies the width of the output field.

  - Default: 9
  - Range: 4–200

### Example

This example uses the en_US locale option.
### NLDATMSw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as a date in the form, MM/DD/YYYY.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

**NLDATMSw.**

### Syntax Description

**w**

Specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 19
- **Range:** 9–200

### Details

NLDATMS outputs the date in a short-uniform pattern with number and delimiters, such as MM/DD/YYYY hh:mm:ss.

### Example

This example uses the date November 19, 2012.
Statements | Results
---|---
data _null_;
dt = datetime();
dy = date();
put "+--- NLDATMS min=9 default=19 max=200 ---+
   dt nldatms.;
   dt nldatms10.;
   dt nldatms19.;
   dt nldatms200.;
run;

NLDATMTMw. Format

Converts the time portion of a SAS datetime value to the time-of-day value of the specified locale, and then writes the value as a time of day.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLDATMTMw.

Syntax Description

w

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16–200</td>
</tr>
</tbody>
</table>

Comparisons

The NLDATMTMw. format is similar to the TODw. format except that the NLDATMTMw. format is locale-specific.

Example

These examples use the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003.

Statements | Results
---|---
options locale=English_UnitedStates; put event nldatmtm.; 12:39:43
Statements | Results
--- | ---
options locale=German_Germany; | 12.39 Uhr
put event nldatmtm.; | 

See Also

Formats:
- “NLDATMw. Format” on page 151
- “NLDATMAPw. Format” on page 152
- “NLDATMw. Format” on page 163

NLDATMTZ\(w\). Format

Converts the time portion of the SAS datetime of the locale to the time of day and time zone.

Category: Date and Time

Alignment: Left

Syntax

NLDATMTZ\(w\).

Syntax Description

\(w\)

specifies the width of the output field.

Default 32

Range 16–200

Example

This example uses the current datetime value.

Statements | Result
--- | ---
options locale=fr_FR; | x=10 h 40 -0400
data test;
x=datetime();
put x=nldatmtz.;
run; |
NLDATMWw. Format

Converts SAS datetime values to the locale sensitive datetime string as the day of the week and the datetime.

Category: Date and Time
Alignment: Left

Syntax

NLDATMWw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the day of week and datetime to fit the format width.

Default: 41
Range: 16–200

Details

The NLDATMW format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 79 for information about recommended widths for locale and encoding combinations. You might need to use the suggested width for the NLDATMW format.

Comparisons

The NLDATMWw. format is similar to the TWMDYWw. format except that the NLDATMWw. format is locale-specific.

Example

These examples use the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----+----2----+----3</td>
<td></td>
</tr>
<tr>
<td>------+--------+--------+--------</td>
<td></td>
</tr>
<tr>
<td>------+--------+--------+--------</td>
<td></td>
</tr>
<tr>
<td>------+--------+--------+--------</td>
<td></td>
</tr>
</tbody>
</table>
### See Also

**Formats:**
- “NLDATMw. Format” on page 151
- “NLDATMAPw. Format” on page 152
- “NLDATMTMw. Format” on page 161

---

### NLDATMWNw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as the day of the week.

**Category:** Date and Time  
**Alignment:** Left

#### Syntax

```
NLDATMWNw.
```

#### Syntax Description

- `w` specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>4–200</td>
</tr>
</tbody>
</table>
Example

This example writes the SAS datetime value as a day of the week.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMWN min=4 default=9 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Monday</td>
</tr>
<tr>
<td>dy = date();</td>
<td>Mon</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMWN min=4 default=9 max=200 ---+&quot;;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn.;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn4.;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn9.;</td>
<td>Monday</td>
</tr>
<tr>
<td>put dt nldatmwn200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMWZw. Format

Converts SAS date values of the specified locale to a day-of-week, datetime, and time zone value.

- **Category:** Date and Time
- **Alignment:** Left

Syntax

NLDATMWZw.

**Syntax Description**

\(w\)

- specifies the width of the output field. If necessary, SAS abbreviates the day of week and datetime to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>16–200</td>
</tr>
</tbody>
</table>

Details

The NLDATMWZ format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 79 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Example

This example uses the current datetime value.
NLDATMYMw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the month.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

NLDATMYMw.

### Syntax Description

- `w` specifies the width of the output field.

  **Default** 16  
  **Range** 6–200

### Example

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=vendredi 18 mars 2011 10 h 40 -0400</td>
<td>x=vendredi 18 mars 2011 10 h 40 -0400</td>
</tr>
</tbody>
</table>

### See Also

Format:
NLDATMYMLw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.

Category: Date and Time
Alignment: Left

Syntax

NLDATMYMLw.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default 14
Range 5–200

Details

NLDATMYML outputs the date in a long-uniform pattern with full length of the month and year, such as November 2012.

Example

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMYML min=5 default=14 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>November 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/2012</td>
</tr>
<tr>
<td>put &quot;*---- NLDATMYML min=5 default=14 max=200 ----&quot;;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatmyml.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dt nldatmyml7.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dt nldatmyml11.;</td>
<td>November 2012</td>
</tr>
<tr>
<td>put dt nldatmyml14.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyml200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
**NLDATMYMMw. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the month and the year.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

NLDATMYMMw.

**Syntax Description**

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

- **Default:** 11
- **Range:** 5–200

**Details**

NLDATMYMM outputs the date in a medium-uniform pattern with abbreviations of the month and year, such as Nov 2012.

**Example**

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+---- NLDATMYMM min=5 default=11 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/2012</td>
</tr>
<tr>
<td>put &quot;**** NLDATMYMM min=5 default=11 max=200 ****&quot;;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymmm.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymm7.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymm11.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>put dt nldatymm200.;</td>
<td>Nov 2012</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLDATMYMSw. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the month and year with numbers and delimiters.

**Category:** Date and Time
Syntax

NLDATMYMS\(w\).

Syntax Description

\(w\)

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default

7

Range

5–200

Details

NLDATMYMS outputs the date in a short-uniform pattern with numbers and delimiters for the month and year, such as 11/2012.

Example

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMYMS min=5 default=7 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>11/2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>11/12</td>
</tr>
<tr>
<td>put &quot;++- NLDATMYMS min=5 default=7</td>
<td></td>
</tr>
<tr>
<td>max=200 ----&quot;;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dt nldatmyms.;</td>
<td>11/2012</td>
</tr>
<tr>
<td>put dt nldatmyms5.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyms7.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyms200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYQw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.

Category: Date and Time

Alignment: Left

Syntax

NLDATMYQw.
Syntax Description

\[ w \]

specifies the width of the output field.

Default 16

Range 4–200

Details

The NLDATMYQ format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 79 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Example

This example uses the fr_FR locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td>+--- NLDATMYQ min=4 default=16 max=200 ---+</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>16 T3 08</td>
</tr>
<tr>
<td>dy=today();</td>
<td>4 ****</td>
</tr>
<tr>
<td>dt=datetime();</td>
<td>14 T3 08</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMYQ min=4 default=16 max=200 ---+&quot;;</td>
<td>32 3e trimestre 2008</td>
</tr>
<tr>
<td>put '16' +5 dt nldatmyq.;</td>
<td>200 3e trimestre 2008</td>
</tr>
<tr>
<td>put '4' +5 dt nldatmyq4.;</td>
<td></td>
</tr>
<tr>
<td>put '14' +5 dt nldatmyq14.;</td>
<td></td>
</tr>
<tr>
<td>put '32' +5 dt nldatmyq32.;</td>
<td></td>
</tr>
<tr>
<td>put '200' +5 dt nldatmyq200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYQLw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter value (1–4) and the year.

Category: Date and Time

Alignment: Left

Syntax

\[ \text{NLDATMYQLw.} \]
Syntax Description

\texttt{w}

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default \hspace{1cm} 18

Range \hspace{1cm} 4–200

Details

NLDATMYQL outputs the date in a long uniform pattern in full length of the year’s quarter and then the year, such as 4th quarter 2012.

Example

The following example uses the date of November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+---- NLDATMYQL min=4 default=18 max=200 ----+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>dy = date();</td>
<td>2012/4</td>
</tr>
<tr>
<td>put &quot;+---- NLDATMYQL min=4 default=18 max=200 ----&quot;;</td>
<td>Q4 2012</td>
</tr>
<tr>
<td>put dt nldatmyql.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>put dt nldatmyql6.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>put dt nldatmyql7.;</td>
<td>4th quarter 2012</td>
</tr>
<tr>
<td>put dt nldatmyql18.;</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyql18.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYQMw. Format

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year’s quarter (1–4) and then the year.

\begin{itemize}
  \item Category: Date and Time
  \item Alignment: Left
\end{itemize}

Syntax

\texttt{NLDATMYQMw}.

Syntax Description

\texttt{w}

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default \hspace{1cm} 7
Range  4–200

**Details**

NLDATMYQM outputs the date in a medium uniform pattern of the year’s quarter and then the year, such as Q4 2012.

**Example**

The following example uses the date of November 19, 2012.

```
data _null_;  
  dt = datetime();  
  dy = date();  
  put "+--- NLDATMYQM min=4 default=7 max=200 ---+";  
  put dt nldatmyqm.;  
  put dt nldatmyqm6.;  
  put dt nldatmyqm7.;  
  put dt nldatmyqm200.;  
run;```

---

**NLDATMYQS**

**w. Format**

Converts a SAS date value to the date string of the specified locale, and then writes the date value as the year and the quarter (1-4) using numbers and delimiters.

**Category:**  Date and Time

**Alignment:**  Left

**Syntax**

NLDATMYQS**w.**

**Syntax Description**

**w**

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

**Default**  6

**Range**  4–200

**Details**

NLDATMYQS outputs the date in a short-uniform pattern of the year and year’s quarter value using numbers and delimiters, such as 2012/4.
Example

The following example uses the date November 19, 2012.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>+--- NLDATMYQS min=4 default=6 max=200 ---+</td>
</tr>
<tr>
<td>dt = datetime();</td>
<td>2012/4</td>
</tr>
<tr>
<td>dy = date();</td>
<td>12/4</td>
</tr>
<tr>
<td>put &quot;+--- NLDATMYQS min=4 default=6 max=200 ---+&quot;;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs4.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs6.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>put dt nldatmyqs200.;</td>
<td>2012/4</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMYRw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLDATMYRw.

Syntax Description

w

specifies the width of the output field.

Default 16

Range 2–200

Example

This example uses the en_US locale option.
Statements

```sas
options locale=fr_FR;
data _null_;  
dy=today();  
dt=datetime();  
put "+--- NLDATMYR min=2 default=16 
max=200 ---+";  
put dt nldatmyr.;  
put dt nldatmyr2.;  
put dt nldatmyr32.;  
put dt nldatmyr200.;  
run;
```

Results

```
+--- NLDATMYR min=2 default=16
max=200 ---+
2008
```

### NLDATMYWw Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.

**Category:** Date and Time  
**Alignment:** Left

---

### Syntax

NLDATMYWw.

---

### Syntax Description

`w`

specifies the width of the output field.

**Default:** 16  
**Range:** 5–200

---

### Example

This example uses the fr_FR locale option.
**Statements**

options locale=fr_FR;
data _null_;  
  dy=today();  
  dt=datetime();  
  put "+--- NLDATMYW min=5 default=16 max=200 ---+";  
  put ' 16' +5 dt nldatmyw.;  
  put ' 5' +5 dt nldatmyw5.;  
  put ' 8' +5 dt nldatmyw8.;  
  put ' 32' +5 dt nldatmyw32.;  
  put '200' +5 dt nldatmyw200.;  
run;

---

## NLDATMZw. Format

Converts SAS datetime values to the locale-sensitive datetime string as time zone and datetime.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

NLDATMZw.

### Syntax Description

- **w** specifies the width of the output field.

  **Default** 40  
  **Range** 16–200

### Example

This example uses the current datetime value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
</table>
| options locale=fr_FR; data test;  
x=datetime();  
put x=nldatmz.;  
run; | x=18 mars 2011 10 h 40 -0400 |
NLMNIAEDw.d Format

Writes the monetary format of the international expression for the United Arab Emirates.

Category: Numeric
Alignment: Left

Syntax

NLMNIAEDw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 3
Range 0–28

Example

In the following example, the LOCALE= system option is set to English UnitedStates.

x=put(-1234.56789,nlmniaed32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AED1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>S-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLAE Dx.d Format” on page 213
**NLMNIAUDw.d Format**

Writes the monetary format of the international expression for Australia.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNIAUD}w.d \]

**Syntax Description**

- \( w \)
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- \( d \)
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789,\text{nlmniaud32.2}); \]
\[ y=\text{put}(-1234.56789,\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{put } x=; )</td>
<td>\text{(AUD}1,234.57)</td>
</tr>
<tr>
<td>( \text{put } y=; )</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNLAUDw.d Format” on page 214
NLMNIBGNw.d Format

Writes the monetary format of the international expression for Bulgaria.

**Category:** Numeric

**Alignment:** Left

### Syntax

\[ \text{NLMNIBGN}w.d \]

### Syntax Description

- \( w \) specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- \( d \) specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 2
  - **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{put} (-1234.56789, \text{nlmnbgn32.2}) \\
y &= \text{put} (-1234.56789, \text{dollar32.2})
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = )</td>
<td>( \text{BGN1,234.57} )</td>
</tr>
<tr>
<td>put ( y = )</td>
<td>(-1,234.57)</td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLMNLBGNw.d Format” on page 215
NLMNIBRLw.d Format

Writes the monetary format of the international expression for Brazil.

**Category:** Numeric

**Alignment:** Left

### Syntax

**NLMNIBRLw.d**

### Syntax Description

**w**

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

**d**

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnibrl32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BRL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLBRLw.d Format” on page 216
**NLMNICADw.d Format**

Writes the monetary format of the international expression for Canada.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

NLMNICADw.d

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnicad32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CAD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “NLMNLCADw.d Format” on page 217
NLMNICHFw.d Format

Writes the monetary format of the international expression for Liechtenstein and Switzerland.

**Category:** Numeric  
**Alignment:** Left

### Syntax

NLMNICHFw.d

### Syntax Description

**w**  
Specifies the width of the output field.

- **Default:** 12  
- **Range:** 8–32

**d**  
Specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2  
- **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnichf32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-------------------</td>
</tr>
<tr>
<td>put x=;</td>
<td>(CHF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

Format:

- “NLMNLCHFw.d Format” on page 218
**NLMNICNYw.d Format**

Writes the monetary format of the international expression for China.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNICNYw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default**: 12
- **Range**: 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default**: 02
- **Range**: 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```r
x=put(-1234.56789,nlmnicny32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CNY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLCTNYw.d Format” on page 219
**NLMNICZKw.d Format**

Writes the monetary format of the international expression for the Czech Republic.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNICZKw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 12  
- **Range:** 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 4  
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniczk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CZK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLNCZKw.d Format” on page 220
NLMNIDKKw.d Format

Writes the monetary format of the international expression for Denmark, Faroe Island, and Greenland.

**Category:** Numeric  
**Alignment:** Left

### Syntax

```
NLMNIDKKw.d
```

### Syntax Description

**w**
- Specifies the width of the output field.  
  - **Default:** 12  
  - **Range:** 8–32

**d**
- Specifies the number of digits to the right of the decimal point in the numeric value.  
  - **Default:** 2  
  - **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnidkk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=</code>;</td>
<td>[DKK1,234.57]</td>
</tr>
<tr>
<td><code>put y=</code>;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLMNLDKKw.d Format” on page 221
NLMNIEEKw.d Format

Writes the monetary format of the international expression for Estonia.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNIEEKw.d

**Syntax Description**

`w`  
specifies the width of the output field.  
Default: 12  
Range: 8–32

`d`  
specifies the number of digits to the right of the decimal point in the numeric value.  
Default: 4  
Range: 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnieek32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EEK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**  
- “NLMNLEEKw.d Format” on page 222
**NLMNIEGPw.d Format**

Writes the monetary format of the international expression for Egypt.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\texttt{NLMNIEGPw.d}  

### Syntax Description

- **\texttt{w}**
  - Specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- **\texttt{d}**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 3
  - **Range:** 0–28

### Example

In the following example, the \texttt{LOCALE=} system option is set to English_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmniegp32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \texttt{x=};</td>
<td>(EGP1,234.57)</td>
</tr>
<tr>
<td>put \texttt{y=};</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

- **Format:**
  - “NLMNLEGw.d Format” on page 223
**NLMNIEURw.d Format**

Writes the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

**Category:** Numeric  
**Alignment:** Left

---

### Syntax

\[ \text{NLMNIEURw.d} \]

### Syntax Description

- **w**  
  specifies the width of the output field.  
  - Default: 12  
  - Range: 8–32

- **d**  
  specifies the number of digits to the right of the decimal point in the numeric value.  
  - Default: 2  
  - Range: 0–28

---

### Example

In the following example, the `LOCALE=` system option is set to `Locale=German_Germany`.

\[
\begin{align*}
\text{x=put(-1234.56789,nlmnieur32.2);} \\
\text{y=put(-1234.56789,nlmnieur32.2);} \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>x=-1.234,57 EUR</td>
</tr>
<tr>
<td>put y=;</td>
<td>y=-1.234,57 €</td>
</tr>
</tbody>
</table>

---

### See Also

**Format:**

- “NLMNLEURw.d Format” on page 224
**NLMNIGBPw.d Format**

Writes the monetary format of the international expression for the United Kingdom.

**Category:** Numeric

**Alignment:** Left

### Syntax

NLMNIGBPw.d

### Syntax Description

**w**

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

**d**

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnigbp32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(GBP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLGBPw.d Format” on page 225
NLMNIHKDw.d Format

Writes the monetary format of the international expression for Hong Kong.

Category: Numeric
Alignment: Left

Syntax

NLMNIHKDw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnihkd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>$-1,234.57</td>
</tr>
<tr>
<td>put y=;</td>
<td>HKD1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLHKDw.d Format” on page 226
NLMNIHRKw.d Format

 Writes the monetary format of the international expression for Croatia.

  | Category: Numeric |
  | Alignment: Left |

Syntax

NLMNIHRKw.d

Syntax Description

w
  specifies the width of the output field.

  Default 12
  Range 8–32

d
  specifies the number of digits to the right of the decimal point in the numeric value.

  Default 2
  Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnihrk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HRK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLHRKw.d Format” on page 227
NLMNIHUFw.d Format

Writers the monetary format of the international expression for Hungary.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[ \text{NLMNIHUF}w.d \]

**Syntax Description**

- **\( w \)** specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- **\( d \)** specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 2
  - **Range:** 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{put}(-1234.56789, \text{nlmnihuf}32.2);
y=\text{put}(-1234.56789, \text{dollar}32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HUF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNLHUFw.d Format” on page 228
**NLMNIIDRw.d Format**

Writes the monetary format of the international expression for Indonesia.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

`NLMNIIDRw.d`

**Syntax Description**

`w`

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

---

**Example**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmniidr32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(IDR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- [“NLMNLIDRw.d Format” on page 229](#)
NLMNIILSw.d Format

Writes the monetary format of the international expression for Israel.

**Category:** Numeric
**Alignment:** Left

### Syntax

\[ \text{NLMNIILS}w.d \]

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 4
  - **Range:** 0–28

### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

\[ \begin{align*}
x &= \text{put}(-1234.56789, \text{nlmniils}32.2) \; ; \\
y &= \text{put}(-1234.56789, \text{dollar}32.2) \\
\end{align*} \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ILS1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLILSw.d Format” on page 230
NLMNIINRw.d Format

Writes the monetary format of the international expression for India.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNIINRw.d

**Syntax Description**

w

specifies the width of the output field.

- Default: 12
- Range: 8–32

d

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniinr32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(INR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

Format:

- “NLMNLINRw.d Format” on page 231
### NLMNIJPYw.d Format

Writes the monetary format of the international expression for Japan.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

`NLMNIJPYw.d`

**Syntax Description**

- `w` specifies the width of the output field.  
  - Default: 12  
  - Range: 8–32

- `d` specifies the number of digits to the right of the decimal point in the numeric value.  
  - Default: 0  
  - Range: 0–28

**Example**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.  

```plaintext
x=put(-1234.56789,nlmnijpy32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>[JPY1,234.57]</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Format:  
- “NLMNLJYPYw.d Format” on page 232
NLMNIKRWw.d Format

Writes the monetary format of the international expression for South Korea.

Category: Numeric
Alignment: Left

Syntax

NLMNIKRWw.d

Syntax Description

\( w \)
  specifies the width of the output field.
  
  Default 12
  Range 8–32

\( d \)
  specifies the number of digits to the right of the decimal point in the numeric value.
  
  Default 0
  Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmnikrw}32.2); \\
y = \text{put}(-1234.56789, \text{dollar}32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(KRW1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLKRWw.d Format” on page 233
**NLMNILTLw.d Format**

Writes the monetary format of the international expression for Lithuania.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

`NLMNILTLw.d`

**Syntax Description**

`w`  
specifies the width of the output field.  
- **Default:** 12  
- **Range:** 8–32

`d`  
specifies the number of digits to the right of the decimal point in the numeric value.  
- **Default:** 4  
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniltl32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LTL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLLTLw.d Format” on page 234
NLMNILVLw.d Format

Writes the monetary format of the international expression for Latvia.

Category: Numeric
Alignment: Left

Syntax

NLMNILVLw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 4
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnilvl32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LVL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLLVLw.d Format” on page 235
**NLMNIMOPw.d Format**

Writes the monetary format of the international expression for Macau.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNIMOPw}.d \]

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmnimop32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MOP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

### See Also

- "NLMNLMOPw.d Format" on page 236
NLMNIMXNw.d Format

Writes the monetary format of the international expression for Mexico.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNIMXNw,d

**Syntax Description**

\( \text{w} \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( \text{d} \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789,\text{nlmnimxn32.2}); \]
\[ y=\text{put}(-1234.56789,\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MXN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLMXNw.d Format” on page 237
**NLMNIMYRw.d Format**

Writes the monetary format of the international expression for Malaysia.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\texttt{NLMNIMYRw.d}

**Syntax Description**

\textit{w}

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

---

**Example**

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
\texttt{x=put(-1234.56789,nlmnimy32.2);}
\texttt{y=put(-1234.56789,dollar32.2);}
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{x=};</td>
<td>(MYR1,234.57)</td>
</tr>
<tr>
<td>\texttt{y=};</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- \textit{“NLMNLMYRw.d Format” on page 238}
NLMNINOKw.d Format

Writes the monetary format of the international expression for Norway.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNINOKw.d

**Syntax Description**

-\( \text{w} \)
  - specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

-\( \text{d} \)
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 2
  - **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmninok32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NOK, 1234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

- **Format:**
  - “NLMNLNOKw.d Format” on page 239
NLMNINZDw.d Format

Writes the monetary format of the international expression for New Zealand.

Category: Numeric
Alignment: Left

Syntax

NLMNINZDw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 12
Range 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmninzd}32.2); \\
y &= \text{put}(-1234.56789, \text{dollar}32.2);
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NZD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNINZDw.d Format” on page 240
NLMNIPLNw.d Format

Writes the monetary format of the international expression for Poland.

Category: Numeric
Alignment: Left

Syntax

NLMNIPLNw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English UnitedStates.

x=put(-1234.56789,nlmnipln32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(PLN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLPLNw.d Format” on page 241
NLMNIRUB\(w.d\) Format

Writes the monetary format of the international expression for Russia.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[ \text{NLMNIRUB} \(w.d\) \]

### Syntax Description

**\(w\)**

- Specifies the width of the output field.
- **Default:** 12  
- **Range:** 8–32

**\(d\)**

- Specifies the number of digits to the right of the decimal point in the numeric value.
- **Default:** 2  
- **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{put}(\text{-1234.56789}, \text{nlmniurub}32.2); \\
y=\text{put}(\text{-1234.56789}, \text{dollar}32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=;)</td>
<td>(\text{(RUB1,234.57)})</td>
</tr>
<tr>
<td>put (y=;)</td>
<td>($-1,234.57)</td>
</tr>
</tbody>
</table>

### See Also

**Format:**  
- “NLMNLRUB\(w.d\) Format” on page 242
NLMNISEKw.d Format

Writes the monetary format of the international expression for Sweden.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNISEKw.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 12
Range: 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default: 2
Range: 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=put(-1234.56789, nlmnisek32.2);
y=put(-1234.56789, dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+---+---+---+---+</td>
<td></td>
</tr>
<tr>
<td>\texttt{put x=;}</td>
<td>{SEK1,234.57}</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLSEKw.d Format” on page 243
NLMNISGDw.d Format

Writes the monetary format of the international expression for Singapore.

Category: Numeric
Alignment: Left

Syntax

NLMNISGDw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnisgd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SGD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLSGDw.d Format” on page 244
**NLMNITHBw.d Format**

Writes the monetary format of the international expression for Thailand.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

\[ \text{NLMNITHB}w.d \]

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 12  
- **Range:** 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2  
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.  
\[
x = \text{put}(-1234.56789, \text{nlmnithb}32.2); \]
\[
y = \text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

---

**See Also**

Format:

- “NLMNLTHBw.d Format” on page 245
NLMNITRYw.d Format

Writes the monetary format of the international expression for Turkey.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNITRYw.d

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 4
  - Range: 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnitry32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TRY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNLTRYw.d Format” on page 246
NLMNITWDw.d Format

Writes the monetary format of the international expression for Taiwan.

Category: Numeric
Alignment: Left

Syntax

NLMNITWDw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnitwd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TWD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNLTW Dw.d Format” on page 247
**NLMNIUSDw.d Format**

Writes the monetary format of the international expression for Puerto Rico and the United States.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

NLMNIUSD\textit{w.d}

### Syntax Description

\textit{w}

specifies the width of the output field.

- **Default:** 912
- **Range:** 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

### Example

In the following example, the \texttt{LOCALE=} system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniusd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \texttt{x=;}</td>
<td>(USD1,234.57)</td>
</tr>
<tr>
<td>put \texttt{y=;}</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLUSDw.d Format” on page 248
NLMNIZARw.d Format

Writes the monetary format of the international expression for South Africa.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLMNIZAR\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 12

Range 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2

Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmizar32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>{ZAR1,234.57}</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNLZARw.d Format” on page 249
NLMNLAEDx.d Format

Writes the monetary format of the local expression for the United Arab Emirates.

Category: Numeric
Alignment: Left

Syntax

NLMNLAEDw.d

Syntax Description

w

specifies the width of the output field.

Default 12
Range 8–32

d

specifies the number of digits to the right of the decimal point in the numeric value.

Default 3
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlaed32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AED1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIAEDw.d Format” on page 176
NLMNLAUDw.d Format

Writes the monetary format of the local expression for Australia.

**Category:** Numeric

**Alignment:** Left

### Syntax

\[ \text{NLMNLAUD}w.d \]

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 2
  - **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlaud32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AU$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

- “NLMNIAUDw.d Format” on page 177
NLMNLBGNw.d Format

Writes the monetary format of the local expression for Bulgaria.

Category: Numeric
Alignment: Left

Syntax

NLMNLBGNw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlbgn32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BGN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNIBGNw.d Format” on page 178
NLMNLBRLw.d Format

 Writes the monetary format of the local expression for Brazil.

<table>
<thead>
<tr>
<th>Category: Num</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLMNLBRLw.d

Syntax Description

\(w\)

specifies the width of the output field.

| Default | 12 |
| Range   | 8–32 |

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

| Default | 2 |
| Range   | 0–28 |

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlbrl32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(R$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIBRLw.d Format” on page 179
NLMNLCADw.d Format

Writes the monetary format of the local expression for Canada.

Category: Numeric
Alignment: Left

Syntax

NLMNLCADw.d

Syntax Description

w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlcad32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+----------</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>(CA$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNICADw.d Format” on page 180
NLMNLCHFw.d Format

Writes the monetary format of the local expression for Liechtenstein and Switzerland.

**Category:** Numeric  
**Alignment:** Left

### Syntax

NLMNLCHFw.d

### Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlchf32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>SFr.1,234.57</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNICHFw.d Format” on page 181
NLMNLCNYw.d Format

W rites the monetary format of the local expression for China.

Category: Numeric
Alignment: Left

Syntax

NLMNLCNYw.d

Syntax Description

w
specifies the width of the output field.
Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.
Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlcny32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(RMB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNICNYw.d Format” on page 182
**NLMNLCZKw.d Format**

 Writes the monetary format of the local expression for the Czech Republic.

**Category:** Numeric  
**Alignment:** Left

### Syntax

NLMNLCZKw.d

### Syntax Description

**w**  
specifies the width of the output field.  
- **Default:** 12  
- **Range:** 8–32

**d**  
specifies the number of digits to the right of the decimal point in the numeric value.  
- **Default:** 4  
- **Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlczk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---+-----+---+----</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>(CZK,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Format:**  
- “NLMNICZKw.d Format” on page 183
NLMNLDKKw.d Format

Wrote the monetary format of the local expression for Denmark, Faroe Island, and Greenland.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNLDKKw.d

**Syntax Description**

\( \text{w} \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( \text{d} \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmnldkk32.2});
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “NLMNIDKKw.d Format” on page 184
NLMNLEEKw.d Format

Writes the monetary format of the local expression for Estonia.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

NLMNLEEKw.d

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 4
  - Range: 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnleek32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “NLMNIEEKw.d Format” on page 185
**NLMNLEGp.d Format**

Writes the monetary format of the local expression for Egypt.

- **Category:** Numeric
- **Alignment:** Left

---

**Syntax**

*NLMNLEGp.d*

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 3
  - Range: 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlegp32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EGP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

- “NLMNIEGPw.d Format” on page 186
**NLMNLEURw.d Format**

Writes the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

NLMNLEURw.d

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

**Example**

In the following example, the LOCALE= system option is set to German_Germany.

```plaintext
x=put(-1234.56789,nlmnieur32.2);
y=put(-1234.56789,nlmnieur32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>x=-1.234,57 EUR</td>
</tr>
<tr>
<td>put y=;</td>
<td>y=-1.234,57 €</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNIEURw.d Format” on page 187
NLMNLGBPw.d Format

Writes the monetary format of the local expression for the United Kingdom.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNLGBPw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlbp32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(£1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

Format:

- “NLMNIGBPw.d Format” on page 188
NLMNLHKDw.d Format

Writes the monetary format of the local expression for Hong Kong.

Category: Numeric
Alignment: Left

Syntax

NLMNLHKDw.d

Syntax Description

w
  specifies the width of the output field.
  Default 12
  Range 8–32

d
  specifies the number of digits to the right of the decimal point in the numeric value.
  Default 2
  Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlhd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>HK$1,234.57</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
• “NLMNIHKDw.d Format” on page 189
NLMNLHRKw.d Format

Writes the monetary format of the local expression for Croatia.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLHRKw.d

**Syntax Description**

`w`

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlhrk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Kn1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “NLMNIHRKw.d Format” on page 190
NLMNLHUFw.d Format

 Writes the monetary format of the local expression for Hungary.

 Category: Numeric
 Alignment: Left

Syntax

NLMNLHUFw.d

Syntax Description

w
  specifies the width of the output field.
  Default 12
  Range  8–32

 d
  specifies the number of digits to the right of the decimal point in the numeric value.
  Default 2
  Range  0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789, nlmnlhuf32.2);
y=put(-1234.56789, dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Ft1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
  • “NLMNIHUFw.d Format” on page 191
NLMNLIDRw.d Format

Writes the monetary format of the local expression for Indonesia.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLIDRw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlmnlidr32.2}); \]
\[ y = \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Rp1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNIIDRw.d Format” on page 192
NLMNLILSw.d Format

Writes the monetary format of the local expression for Israel.

Category: Numeric
Alignment: Left

Syntax

NLMNLILSw.d

Syntax Description

w

specifies the width of the output field.

Default 12
Range 8–32

d

specifies the number of digits to the right of the decimal point in the numeric value.

Default 4
Range 0–28

Example

In the following example, the LOCATE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlils32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ILS1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIIILSw.d Format” on page 193
**NLMNLINRw.d Format**

Wrote the monetary format of the local expression for India.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNLINRw.d} \]

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 2
  - **Range:** 0–28

### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnlinr}32.2) ; \\
y &= \text{put}(-1234.56789, \text{dollar}32.2) ;
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(INR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

### See Also

- “NLMNIINRw.d Format” on page 194
NLMNLJPYw.d Format

Writes the monetary format of the international expression for Japan.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\texttt{NLMNLJPYw.d}

**Syntax Description**

\texttt{w}

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 0
- **Range:** 0–28

**Example**

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnljpy32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>(JPY1,234.57)</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “NLMNIJPYw.d Format” on page 195
NLMNLKRWw.d Format

Writers the monetary format of the local expression for South Korea.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLKRWw.d}

\textbf{Syntax Description}

\texttt{w}

specifies the width of the output field.

Default 12
Range 8–32

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 0
Range 0–28

Example

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.

\begin{verbatim}
\texttt{x=put(-1234.56789,nlmnlkrw32.2)};
\texttt{y=put(-1234.56789,dollar32.2)};
\end{verbatim}

\begin{tabular}{lc}
\hline
Statements & Results \\
\hline
\texttt{put x=;} & (KRW1,234.57) \\
\texttt{put y=;} & $-1,234.57$ \\
\hline
\end{tabular}

See Also

Format:

\begin{itemize}
\item “NLMNIKRWw.d Format” on page 196
\end{itemize}
NLMNLLTL\textit{w.d} Format

Writes the monetary format of the local expression for Lithuania.

Category: Numeric
Alignment: Left

Syntax

\texttt{NLMNLLTL\textit{w.d}}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 12
Range 8–32

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Default 4
Range 0–28

Example

In the following example, the LOCALE= system option is set to English\_UnitedStates.

\texttt{x=put(-1234.56789,nlmnltl32.2)};
\texttt{y=put(-1234.56789,dollar32.2)};

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LT1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Format:

\begin{itemize}
\item “NLMNILTL\textit{w.d} Format” on page 197
\end{itemize}
**NLMNLLVLw.d Format**

Writes the monetary format of the local expression for Latvia.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

\[ \text{NLMNLLVL}w.d \]

### Syntax Description

- **w**
  - specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 4
  - **Range:** 0–28

### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=put(-1234.56789,nlmnllvl32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(Ls1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNILVLw.d Format” on page 198
NLMNLMOPw.d Format

Writes the monetary format of the local expression for Macau.

- **Category:** Numeric
- **Alignment:** Left

Syntax

NLMNLMOPw.d

**Syntax Description**

\( \text{w} \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( \text{d} \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlmop32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put(-1234.56789,nlmnlmop32.2);</td>
<td>(P1,234.57)</td>
</tr>
<tr>
<td>y=put(-1234.56789,dollar32.2);</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNIMOPw.d Format” on page 199
NLNMNLMXNw.d Format

Writes the monetary format of the local expression for Mexico.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLNMNLMXNw.d

**Syntax Description**

w

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>8–32</td>
</tr>
</tbody>
</table>

d

specifies the number of digits to the right of the decimal point in the numeric value.

<table>
<thead>
<tr>
<th>Default</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>0–28</td>
</tr>
</tbody>
</table>

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmnlmxn32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
</tr>
<tr>
<td>put y=;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(MX$1,234.57)</td>
</tr>
<tr>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**
- “NLMNIMXNw.d Format” on page 200
NLMNLMYRw.d Format

Writes the monetary format of the local expression for Malaysia.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

**Syntax**

\[
\text{NLMNLMYRw.d}
\]

**Syntax Description**

\[w\]

specifies the width of the output field.

- Default: 12
- Range: 8–32

\[d\]

specifies the number of digits to the right of the decimal point in the numeric value.

- Default: 2
- Range: 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{put}(-1234.56789,\text{nlmnlmyr32.2});
y=\text{put}(-1234.56789,\text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(R1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- “NLMNIMYRw.d Format” on page 201
NLMNLNOKw.d Format

Writes the monetary format of the local expression for Norway.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNLNOKw.d

**Syntax Description**

\(w\)

specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>8–32</td>
</tr>
</tbody>
</table>

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0–28</td>
</tr>
</tbody>
</table>

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnl nok32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNINOKw.d Format” on page 202
**NLMNLNZDw.d Format**

Writes the monetary format of the local expression for New Zealand.

**Category:** Numeric  
**Alignment:** Left

---

**Syntax**

NLMNLNZDw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnlnzd32.2}); \\
y &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NZ$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNINZDw.d Format” on page 203
NLMNLPLNw.d Format

W rites the monetary format of the local expression for Poland.

Category: Numeric
Alignment: Left

Syntax

NLMNLPLNw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 12
Range 8–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default 2
Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789,\text{nlmnlpln32.2}) ; \]
\[ y=\text{put}(-1234.56789,\text{dollar32.2}) \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(PLN 1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNIPLNw.d Format” on page 204
NLMNLRUBw.d Format

Writes the monetary format of the local expression for Russia.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNLRUB\(w.d\)

**Syntax Description**

\(w\)

specifies the width of the output field.

- **Default:** 12
- **Range:** 8–32

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 2
- **Range:** 0–28

---

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlrub32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>{RUB1,234.57}</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

---

**See Also**

**Format:**

- “NLMNIRUBw.d Format” on page 205
NLMNLSEKw.d Format

Writes the monetary format of the local expression for Sweden.

**Category:** Numeric  
**Alignment:** Left

**Syntax**

\texttt{NLMNLSEKw.d}

**Syntax Description**

\texttt{w}  
\text{specifies the width of the output field.}  
\text{Default} \quad 12  
\text{Range} \quad 8–32

\texttt{d}  
\text{specifies the number of digits to the right of the decimal point in the numeric value.}  
\text{Default} \quad 2  
\text{Range} \quad 0–28

**Example**

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English_UnitedStates}.

\begin{verbatim}
x=put(-1234.56789,nlmnlsek32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

Format:  
- “NLMISEKw.d Format” on page 206
NLMNLSGDw.d Format

Writes the monetary format of the local expression for Singapore.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

\[ \text{NLMNLSGD}w.d \]

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlsgd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SG$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “NLMNISGDw.d Format” on page 207
NLMNLTHBw.d Format

Writes the monetary format of the local expression for Thailand.

Category: Numeric
Alignment: Left

Syntax

NLMNLTHBw.d

Syntax Description

w
  specifies the width of the output field.
  Default 12
  Range 8–32

d
  specifies the number of digits to the right of the decimal point in the numeric value.
  Default 2
  Range 0–28

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnlthb32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

• “NLMNITHBw.d Format” on page 208
NLMNLTRYw.d Format
Writes the monetary format of the local expression for Turkey.

Category: Numeric
Alignment: Left

Syntax
NLMNLTRYw.d

Syntax Description
w
specifies the width of the output field.

Default 12
Range 8–32

d
specifies the number of digits to the right of the decimal point in the numeric value.

Default 4
Range 0–28

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

x=put(-1234.56789,nlmnltry32.2);  
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(YTL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNITRYw.d Format” on page 209
NLMNLTDWd Format

Writes the monetary format of the local expression for Taiwan.

- **Category:** Numeric
- **Alignment:** Left

## Syntax

NLMNLTDWd

### Syntax Description

**w**
- Specifies the width of the output field.
  - Default: 12
  - Range: 8–32

**d**
- Specifies the number of digits to the right of the decimal point in the numeric value.
  - Default: 2
  - Range: 0–28

## Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnltd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NT$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

## See Also

- “NLMNITWDw.d Format” on page 210
NLMNLUSDw.d Format

Writes the monetary format of the local expression for Puerto Rico and the United States.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

NLMNLUSDw.d

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - **Default:** 12
  - **Range:** 8–32

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.
  - **Default:** 2
  - **Range:** 0–28

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnlusd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(US$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

- "NLMNIUSDw.d Format" on page 211
**NLMNLZARw.d Format**

Writes the monetary format of the local expression for South Africa.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[ \text{NLMNLZAR} \, w, \, d \]

### Syntax Description

**w**  
specifies the width of the output field.  
**Default:** 12  
**Range:** 8–32

**d**  
specifies the number of digits to the right of the decimal point in the numeric value.  
**Default:** 2  
**Range:** 0–28

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmnlzar32.2});
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(R1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

### See Also

**Format:**  
- “NLMNIZARw.d Format” on page 212
NLMNYw.d Format

Writes the monetary format of the local expression in the specified locale using local currency.

**Category:** Numeric

**Alignment:** Left

---

**Syntax**

NLMNYw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

- **Default:** 0
- **Range:** 0–31

**Details**

The NLMNYw.d informat reads integer binary (fixed-point) values, including negative values that are represented in two’s-complement notation. The NLMNYw.d format writes numeric values by using the currency symbol, the thousands separator, and the decimal separator that is used by the locale.

Note: The NLMNYw.d format does not convert currency format. Therefore, the value of the formatted number should equal the currency of the current locale value.

**Comparisons**

The NLMNYw.d and NLMNYIw.d formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYIw.d format uses three-letter international currency codes, such as USD, while NLMNYw.d format uses local currency symbols, such as $.

The NLMNYw.d format is similar to the DOLLARw.d format except that the NLMNYw.d format is locale-specific.

**Example**

In the following example, the `LOCALE=` system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmny32.2);
y=put(-1234.56789,dollar32.2);
```
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>($1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLMNYIw.d Format” on page 251

Informats:
- “NLMNYw.d Informat” on page 526
- “NLMNYIw.d Informat” on page 527

NLMNYIw.d Format

Writers the monetary format of the international expression in the specified locale.

Category: Numeric
Alignment: Left

Syntax

NLMNYIw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Default 0
Range 0–31

Details

The NLMNYIw.d informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMNYIw.d format writes numeric values by using the international currency code, and locale-dependent
thousands and decimal separators. The position of international currency code is also locale dependent.

Note: The NLMNYIw.d format does not convert currency format. Therefore, the value of the formatted number should equal the currency of the current locale value.

Comparisons

The NLMNYw.d and NLMNYIw.d formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYIw.d format uses three-letter international currency codes, such as USD, while NLMNYw.d format uses local currency symbols, such as $.

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789,\text{nlmnyi32.2}); \]
\[ y=\text{put}(-1234.56789,\text{nlmny32.2}); \]
\[ z=\text{put}(-1234.56789,\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x=; )</td>
<td>(USD1,234.57)</td>
</tr>
<tr>
<td>put ( y=; )</td>
<td>($1,234.57)</td>
</tr>
<tr>
<td>put ( z=; )</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNYw.d Format” on page 250

Informats:

- “NLMNYw.d Informat” on page 526
- “NLMNYIw.d Informat” on page 527

NLNUMw.d Format

Writes the numeric format of the local expression in the specified locale.

Category: Numeric

Alignment: Left

Syntax

NLNUMw.d
Syntax Description

\[ w \]

specifies the width of the output field.

Default: 6

Range: 1–32

\[ d \]

specifies to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.

Default: 0

Range: 0–31

Details

The NLNUM\(w,d\) informat reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLNUM\(w,d\) format writes numeric values by using the thousands separator and the decimal separator that is used by the locale.

Comparisons

The NLNUM\(w,d\) format writes the numeric value with locale-dependent thousand and decimal separators. The NLNUMI\(w,d\) format writes the numeric value with a comma (,) as thousands separator and a period (.) as a decimal separator.

If the \(w\) or \(d\) values are not large enough to generate a formatted number, the NLNUM\(w,d\) format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

Example

\[ x=\text{put}(-1234356.7891,\text{nlnum}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{options LOCALE=English_{UnitedStates};} ]</td>
<td>[ -1,234,356.79 ]</td>
</tr>
<tr>
<td>[ \text{put } x=; ]</td>
<td>[ -1,234,356.79 ]</td>
</tr>
<tr>
<td>[ \text{options LOCALE=German_Germany;} ]</td>
<td>[ -1.234 356,79 ]</td>
</tr>
<tr>
<td>[ \text{put } x=; ]</td>
<td>[ -1.234 356,79 ]</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLNUMI\(w,d\) Format” on page 254
Informs:

- “NLNUMw.d Informat” on page 529
- “NLNUMIw.d Informat” on page 530

### NLNUMIw.d Format

Writes the numeric format of the international expression in the specified locale.

**Category:** Numeric  
**Alignment:** Left

#### Syntax

NLNUMI\textit{w.d}

#### Syntax Description

\textit{w}  
- specifies the width of the output field.  
  - Default: 6  
  - Range: 1–32

\textit{d}  
- specifies to divide the number by 10^d. If the data contains decimal points, the \textit{d} value is ignored.  
  - Default: 0  
  - Range: 0–31

#### Details

The `NLNUMI\textit{w.d}` informat reads integer binary (fixed-point) values, including negative values that are represented in two’s-complement notation. The `NLNUMI\textit{w.d}` format writes numeric values by using a comma (,) as thousands separator and a period (.) as a decimal separator for all locales.

#### Comparisons

The `NLNUMI\textit{w.d}` format writes the numeric data of the international expression in the specified locale. The `NLNUMI\textit{w.d}` format writes the numeric value with a comma (,) as thousands separator and a period (.) as a decimal separator.

If the \textit{w} or \textit{d} values are not large enough to generate a formatted number, the `NLNUM\textit{w.d}` format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

#### Example

\begin{verbatim}
x=put(-123456.7891,nlnumi32.2);
\end{verbatim}
### NLPCTw.d Format

Writes percentage data of the local expression in the specified locale.

#### Category:
Numeric

#### Alignment:
Left

#### Syntax

\[ \text{NLPCTw}.d \]

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 6
  - Range: 4–32

- **d**
  - Specifies to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.
  - Default: 0
  - Range: 0–31

### See Also

**Format:**
- “NLNUMw.d Format” on page 252

**Informats:**
- “NLNUMw.d Informat” on page 529
- “NLNUMIw.d Informat” on page 530
Comparisons

The NLPCTW.d format writes percentage data of the local expression in the specified locale. The NLPCTW.d format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTIW.d format writes the percentage value with a comma (,) as thousands separator and a period (.) as a decimal separator.

The NLPCTW.d format is similar to the PERCENTW.d format except the NLPCTW.d format is locale-specific.

Example

```plaintext
x=put(-12.3456789,nlpct32.2);
y=put(-12.3456789,nlpcti32.2);
z=put(-12.3456789,percent32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-----------------</td>
</tr>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>1234.57%</td>
</tr>
<tr>
<td>put z=;</td>
<td>1234.57%</td>
</tr>
<tr>
<td>options LOCALE=German_Germany;</td>
<td>-1.234,57%</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1.234,57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>1234.57%</td>
</tr>
<tr>
<td>put z=;</td>
<td>1234.57%</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLPCTIW.d Format” on page 256

Informats:
- “NLPCTW.d Informat” on page 531
- “NLPCTIW.d Informat” on page 533

NLPCTIW.d Format

Writes percentage data of the international expression in the specified locale.

- **Category:** Numeric
- **Alignment:** Left
Syntax

**NLPCTIw.d**

**Syntax Description**

\( w \)

specifies the width of the output field.

Default 6

Range 4–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal separators, the \( d \) value is ignored.

Default 0

Range 0–31

**Comparisons**

The NLPCTI\( w.d \) format writes percentage data of the international expression in the specified locale. The NLPCT\( w.d \) format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTI\( w.d \) format writes the percentage value with a comma (,) as thousands separator and a period (.) as a decimal separator.

The NLPCT\( w.d \) format is similar to the PERCENT\( w.d \) format except the NLPCT\( w.d \) format is locale-specific.

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ \begin{align*}
x &= \text{put}(-12.3456789, \text{nlpcti32.2}); \\
y &= \text{put}(-12.3456789, \text{percent32.2});
\end{align*} \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>------------------------------</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>[1234.57]</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “NLPCT\( w.d \) Format” on page 255

**Informats:**

- “NLPCT\( w.d \) Informat” on page 531
NLPCTNw.d Format

Produces percentages, using a minus sign for negative values.

**Category:** Numeric  
**Alignment:** Right

### Syntax

NLPCTNw.d

#### Syntax Description

\[ w \]

Specifies the width of the output field.

- **Default:** 6
- **Range:** 4–32

**Tip**
The width of the output field must account for the minus sign ( – ), the percent sign ( % ), and a trailing blank, whether the number is negative or positive.

\[ d \]

Specifies the number of digits to the right of the decimal point in the numeric value. This argument is optional.

- **Range:** 0–31
- **Requirement:** must be less than \( w \)

### Details

The NLPCTNw.d format multiplies negative values by 100, adds a minus sign to the beginning of the value, and adds a percent sign (%) to the end of the formatted value.

### Example

\[ x = -0.02; \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x nlpctn6;</td>
<td>x=-2%</td>
</tr>
<tr>
<td>put x percentn6;</td>
<td>x=-2%</td>
</tr>
</tbody>
</table>
NLPCTPw.d Format

Writes locale-specific numeric values as percentages.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Right</td>
</tr>
</tbody>
</table>

Syntax

NLPCTPw.d

Syntax Description

\(w\)

specifies the width of the output field.

- Default: 6
- Range: 4–32
- Tip: The width of the output field must account for the percent sign (%).

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value. This argument is optional. The thousands separator and decimal symbol for the NLPCTP format is locale-specific.

- Range: 0–31
- Requirement: must be less than \(w\)

Details

The NLPCTPw.d format multiplies values by 100, formats them, and adds a percent sign (%) to the end of the formatted value. The NLPCTPw.d format is similar to the The PERCENTw.d format except that the thousands separator and decimal symbol for the NLPCTPw.d format is locale-specific.

Example

\[x = -0.02;\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x nlpctp6;</td>
<td>−2%</td>
</tr>
<tr>
<td>put x percent6;</td>
<td>( 2%)</td>
</tr>
</tbody>
</table>
NLPVALUEw.d Format

Writes p-values of the local expression in the specified locale.

**Category:** Numeric
**Alignment:** Left

### Syntax

\texttt{NLPVALUEw.d}

### Syntax Description

- \textit{w}
  - Specifies the width of the output field.
  - Default: 6
  - Range: 3–32

- \textit{d}
  - Specifies to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.
  - Default: 4
  - Range: 1–30

### Example

This example uses the german_Germany locale option.

**Statements:**

```plaintext
options locale=german_germany;
data _null_;  
put "+--- nlpvalue min=3 default=6 max=32 ----+";  
x=0.1248;  
put x= +5 x pvalue +5 x nlpvalue.;  
put x= +5 x pvalue1.1 +5 x nlpvalue1.1;  
put x= +5 x pvalue20.2 +5 x nlpvalue20.2;  
put x= +5 x pvalue32.3 +5 x nlpvalue32.3;  
run;
```

**Results:**

```
+--- nlpvalue min=3 default=6 max=32 ----+
 x=0.1248  0.1248   0,1248
 x=0.1248  0.1       0,1
 x=0.1248  0.12      0,12
 x=0.1248  0.125     0,125
```
NLSTRMONw.d Format

W rites the month name in the specified locale.

Category: Numeric
Alignment: Left

Syntax

NLSTRMONw.d

Syntax Description

\(w\)
specifies the width of the output field

Default 20
Range 200-1

\(d\)
specifies the following:

- 00000001: write abbreviated form.
- 00000010: write capitalized form.

Default 0
Range 0-3

Details

The NLSTRMONw.d format writes a SAS value, 1–12 as the name-of-the-month in the specified locale. The following examples use the English_UnitedStates locale.

- 1 = the first month (January)
- 2 = the second month (February)
- 3 = the third month (March)
- 4 = the fourth month (April)
- 5 = the fifth month (May)
- 6 = the sixth month (June)
- 7 = the seventh month (July)
- 8 = the eight month (August)
• 9 = the ninth month (September)
• 10 = the tenth month (October)
• 11 = the eleventh month (November)
• 12 = the twelfth month (December)

Example
This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>monnum = 1; /* January=1, December=12 */</td>
<td>January</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.;</td>
<td>Jan</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.1; /* decimal .1 specified use abbreviation. */</td>
<td>JANUARY</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.2;</td>
<td>JAN</td>
</tr>
<tr>
<td>put monnum NLSTRMON20.3;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLSTRQTRw.d Format**

 Writes a numeric value as the quarter-of-the-year in the specified locale.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

`NLSTRQTRw.d`

**Syntax Description**

- **w** specifies the width of the output field
  - Default 20
  - Range 1–200

- **d** specifies the following:
  - 00000001: write abbreviated form.
  - 00000010: write capitalized form.
  - Default 3
Details
The NLSTRQTRw.d format writes a SAS value, 1–4 as the name-of-the-quarter for the year in the specified locale. The following examples use the English_UnitedStates locale.

• 1 = 1st quarter
• 2 = 2nd quarter
• 3 = 3rd quarter
• 4 = 4th quarter

Example
This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>;</td>
<td>1st quarter</td>
</tr>
<tr>
<td>qtrnum = 1 ; /* January=1, December=12 */</td>
<td>Q1</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20. ;</td>
<td>1ST QUARTER</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.1; /* decimal .1 specified use abbreviation. */</td>
<td>Q1</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.2;</td>
<td></td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.3; run;</td>
<td></td>
</tr>
</tbody>
</table>

NLSTRWKw.d Format
Writes a numeric value as the day-of-the-week in the specified locale.

Category: Numeric
Alignment: Left

Syntax
NLSTRWKw.d

Syntax Description

w
specifies the width of the output field

Default 20
Range 1–200
The NLSTRWKw.d format writes a SAS value, 1–7 as the name-of-the-week in the specified locale. The following examples use the English_UnitedStates locale.

- 1 = First day-of-week (Monday)
- 2 = Second day-of-week (Tuesday)
- 3 = Third day-of-week (Wednesday)
- 4 = Fourth day-of-week (Thursday)
- 5 = Fifth day-of-week (Friday)
- 6 = Sixth day-of-week (Saturday)
- 7 = Seventh day-of-week (Sunday)

### Example

This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>wknun = 1; /* Sunday=1, Saturday=7 */</td>
<td>Sunday</td>
</tr>
<tr>
<td>put wknun NLSTRWK20.;</td>
<td>Sun</td>
</tr>
<tr>
<td>put wknun NLSTRWK20.1; /* decimal .1 specified use abbreviation. */</td>
<td>SUNDAY</td>
</tr>
<tr>
<td>put wknun NLSTRWK20.2;</td>
<td>SUN</td>
</tr>
<tr>
<td>put wknun NLSTRWK20.3;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

### NLTIMAPw Format

Converts a SAS time value to the time value of a specified locale, and then writes the value as a time value with a.m. or p.m. NLTIMAP also converts SAS date-time values.

**Category:** Date and Time

**Alignment:** Left
Syntax

NLTIMAP\textsubscript{w}.

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 10

Range 4–200

Details

The NLTIMAP format might produce inaccurate localized output when using the default width with some encoding and locale combinations because the date and time names are too long. Please refer to Exceptions for Date and Time Default Widths on page 79 for information about recommended widths for locale and encoding combinations. You might need to use the recommended width.

Comparisons

The NLTIMAP\textsubscript{w}. format is similar to the TIMEAMPM\textsubscript{w}. format except that the NLTIMAP\textsubscript{w}. format is locale-specific.

Example

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; put time nltimap.;</td>
<td>4:24:43 PM</td>
</tr>
<tr>
<td>options locale=German_Germany; put time nltimap14.;</td>
<td>4:24:43 nachm</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLTIMEW. Format” on page 265

NLTIMEW. Format

Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value. NLTIME also converts SAS date-time values.
Syntax

NLTIME\(w\).

Syntax Description

\(w\)

specifies the width of the input field.

Default \(20\)

Range \(10–200\)

Comparisons

The \(NLTIME_{w}\) format is similar to the \(TIME_{w}\) format except that the \(NLTIME_{w}\) format is locale-specific.

Example

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>4:24:43</td>
</tr>
<tr>
<td>put time nltime.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>16.24</td>
</tr>
<tr>
<td>put time nltime.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “\(NLTIMAP_{w}\) Format” on page 264

\(\$UCS2B_{w}\) Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 16-bit, UCS2, Unicode encoding.
Syntax
$UCS2Bw

Syntax Description

$w
specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8

Range 2–32767

Details

The $UCS2Bw format writes a character string in big-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

Comparisons

The $UCS2Bw format performs processing that is the opposite of the $UCS2BEw format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>5927</td>
</tr>
<tr>
<td>x = 'Ａ';</td>
<td></td>
</tr>
<tr>
<td>y=put (x,$ucs2b2.);</td>
<td></td>
</tr>
<tr>
<td>put y $hex.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Lw. Format” on page 269
- “$UCS2Xw. Format” on page 271
- “$UTF8Xw. Format” on page 288
- “$UCS2BEw. Format” on page 268

Informats:
$UCS2BEw. Format

Processes a character string that is in big-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

**Category:** Character  
**Alignment:** Left

**Syntax**

$UCS2BEw.

**Syntax Description**

w specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1–32000</td>
</tr>
</tbody>
</table>

**Details**

The $UCS2BEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in big-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

**Comparisons**

The $UCS2BEw. format performs processing that is the opposite of the $UCS2Bw. format.

**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>--------</td>
</tr>
</tbody>
</table>
$\text{UCS2Lw. Format}$

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 16-bit, UCS2, Unicode encoding.

**Category:** Character  
**Alignment:** Left

**Syntax**

$\text{UCS2Lw}$

**Syntax Description**

`w` specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2–32767</td>
</tr>
</tbody>
</table>

**Details**

The $\text{UCS2Lw}$ format writes a character string in little-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

**Comparisons**

The $\text{UCS2Lw}$ format performs processing that is the opposite of the $\text{UCS2LEw}$ format.

---

**Statements**

$x = '592700410042' x;$  
put x $ucs2be.;

**Results**

$xAB$
Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x = '大';</td>
<td>2759</td>
</tr>
<tr>
<td>y=put(x,$ucs2l2.);</td>
<td></td>
</tr>
<tr>
<td>put y $hex.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Bw. Format” on page 266
- “$UCS2LEw. Format” on page 270
- “$UCS2Xw. Format” on page 271
- “$UTF8Xw. Format” on page 288

Informats:
- “$UCS2Bw. Informat” on page 538
- “$UCS2Lw. Informat” on page 540
- “$UCS2LEw. Informat” on page 541
- “$UCS2Xw. Informat” on page 542
- “$UTF8Xw. Informat” on page 557

$UCS2LEw. Format

Processes a character string that is in little-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left

Syntax

$UCS2LEw.
Syntax Description

\( w \)

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default \( 8 \)

Range \( 1-32000 \)

Details

The \$UCS2LEw\$ format writes a character string in the encoding of the current SAS session. It processes character strings that are in little-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

Comparisons

The \$UCS2LEw\$ format performs processing that is the opposite of the \$UCS2Lw\$ format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '275941004200'x; put x $ucs2le.;</td>
<td>AB</td>
</tr>
</tbody>
</table>

See Also

Format:

- “$UCS2Lw. Format” on page 269

Informats:

- “$UCS2Lw. Informat” on page 540
- “$UCS2LEw. Informat” on page 541

$UCS2Xw. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 16-bit, UCS2, Unicode encoding.

Category: Character

Alignment: Left
Syntax
$UCS2Xw.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8
Range 2–32767

Details

The $UCS2Xw. format writes a character string in 16-bit, UCS2 (universal character set code in two octets), Unicode encoding, by using byte order that is native to the operating environment.

Comparisons

The $UCS2Xw. format performs processing that is the opposite of the $UCS2XEw. format. If you are exchanging data within the same operating environment, use the $UCS2Xw. format. If you are exchanging data with a different operating environment, use the $UCS2Bw. format or $UCS2Lw. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '大'; put x $ucs2x2.;</td>
<td>'5927'x (binary) or '2759'x (little endian)</td>
</tr>
</tbody>
</table>

See Also

Formats:

- “$UCS2Bw. Format” on page 266
- “$UCS2XEw. Format” on page 273
- “$UCS2Lw. Format” on page 269
- “$UTF8Xw. Format” on page 288

Informats:

- “$UCS2Bw. Informat” on page 538
- “$UCS2Lw. Informat” on page 540
$UCS2XEw. Format

Processes a character string that is in native-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

**Category:** Character

**Alignment:** Left

**Syntax**

$UCS2XEw.

**Syntax Description**

$w

$w specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

**Default** 8

**Range** 1–32000

**Details**

The $UCS2XEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in native-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

**Comparisons**

The $UCS2XEw. format performs processing that is the opposite of the $UCS2Xw. format.

**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 'e5a4a7'x; /* Japanese '竹' in UTF8 */; put x $utf8xe10.;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-----+-----</td>
</tr>
</tbody>
</table>

$UCS4Bw. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 32-bit, UCS4, Unicode encoding.

Category: Character
Alignment: Left

Syntax
$UCS4Bw:

Syntax Description

\[ w \]

\( w \) specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 4
Range 4–32767

Details

The $UCS4Bw. format writes a character string in big-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

Comparisons

The $UCS4Bw. format performs processing that is the opposite of the $UCS4BEw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.
Statements | Result
---+----
x = '犬'; | '00005927'x (binary)
put x $ucs4b4.;

See Also

Formats:
- “$UCS2Lw. Format” on page 269
- “$UCS2Xw. Format” on page 271
- “$UCS4BEw. Format” on page 275
- “$UCS4Lw. Format” on page 276
- “$UCS4Xw. Format” on page 279
- “$UTF8Xw. Format” on page 288

Informats:
- “$UCS2Bw. Informat” on page 538
- “$UCS2Lw. Informat” on page 540
- “$UCS2Xw. Informat” on page 542
- “$UCS4Bw. Informat” on page 544
- “$UCS4Lw. Informat” on page 545
- “$UCS4Xw. Informat” on page 546
- “$UTF8Xw. Informat” on page 557

$UCS4BEw. Format
Processes a character string that is in big-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left

Syntax
$UCS4BEw.

Syntax Description

`w`
specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.
Default  8
Range   1–32000

Details
The $UCS4BEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in big-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

Comparisons
The $UCS4BEw. format performs processing that is the opposite of the $UCS4Bw. format.

Example
This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '000059270000004100000042'x; put x $ucs4be.;</td>
<td>AB</td>
</tr>
</tbody>
</table>

See Also
Format:
- “$UCS4Bw. Format” on page 274

Informat:
- “$UCS4Bw. Informat” on page 544

$UCS4Lw. Format
Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 32-bit, UCS4, Unicode encoding.

Category: Character
Alignment: Left

Syntax
$UCS4Lw:
**Syntax Description**

\( w \)

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 4

Range 4–32767

**Details**

The `$UCS4Lw` format writes a character string in little-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

**Comparisons**

The `$UCS4Lw` format performs processing that is the opposite of the `$UCS4LEw` format.

**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>2759</td>
</tr>
<tr>
<td>x = '𠮷';</td>
<td></td>
</tr>
<tr>
<td>y=put(x,$ucs4l4.);</td>
<td></td>
</tr>
<tr>
<td>put y $hex.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

- “$UCS2Bw. Format” on page 266
- “$UCS2Xw. Format” on page 271
- “$UCS4Bw. Format” on page 274
- “$UCS4LEw. Format” on page 278
- “$UCS4Xw. Format” on page 279
- “$UTF8Xw. Format” on page 288

**Informats:**

- “$UCS2Bw. Informat” on page 538
$UCS4LEw. Format

Processes a character string that is in little-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

**Category:** Character  
**Alignment:** Left

**Syntax**

$UCS4LEw.

**Syntax Description**

w  

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1–32000</td>
</tr>
</tbody>
</table>

**Details**

The $UCS4LEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in little-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

**Comparisons**

The $UCS4LEw. format performs processing that is the opposite of the $UCS4Lw. format.

**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----</td>
<td>1</td>
</tr>
</tbody>
</table>

---

- “$UCS2Lw. Informat” on page 540
- “$UCS2Xw. Informat” on page 542
- “$UCS4Bw. Informat” on page 544
- “$UCS4Lw. Informat” on page 545
- “$UCS4Xw. Informat” on page 546
- “$UTF8Xw. Informat” on page 557
Statements | Result
---|---
x ='275900004100000042000000'x;
put x $ucs4le.;

See Also

Format:
- “$UCS4Lw. Format” on page 276

Informat:
- “$UCS4Lw. Informat” on page 545

$UCS4Xw. Format
Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 32-bit, UCS4, Unicode encoding.

Category: Character
Alignment: Left

Syntax
$UCS4Xw.

Syntax Description

\[ w \]

\( w \) specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 4
Range 4–32767

Details
The $UCS4Xw. format writes a character string in 32-bit, UCS4 (universal character set code in two octets), Unicode encoding, by using byte order that is native to the operating environment.

Comparisons
The $UCS4Xw. format performs processing that is the opposite of the $UCS4XEw. format. If you are exchanging data within the same operating environment, use the $UCS4Xw. format. If you are exchanging data with a different operating environment, use the $UCS4Bw. format or $UCS4Lw. format.
Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '日本';</td>
<td>'00005927'x (binary) or '27590000'x (little endian)</td>
</tr>
<tr>
<td>put x $ucs4x4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Lw. Format” on page 269
- “$UCS4XEw. Format” on page 280
- “$UCS2Xw. Format” on page 271
- “$UCS4Bw. Format” on page 274
- “$UCS4Lw. Format” on page 276
- “$UTF8Xw. Format” on page 288

Informats:
- “$UCS2Bw. Informat” on page 538
- “$UCS2Lw. Informat” on page 540
- “$UCS2Xw. Informat” on page 542
- “$UCS4Bw. Informat” on page 544
- “$UCS4Bw. Format” on page 274
- “$UCS4Lw. Informat” on page 545
- “$UCS4Xw. Informat” on page 546
- “$UTF8Xw. Informat” on page 557

$UCS4XEw. Format

Processes a character string that is in native-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.

Category: Character
Alignment: Left

Syntax

$UCS4XEw.
Syntax Description

\( w \)

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 8
Range 1–32000

Details

The $UCS4XEw. format writes a character string in the encoding of the current SAS session. It processes character strings that are in native-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

Comparisons

The $UCS4XEw. format performs processing that is the opposite of the $UCS4Xw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '27590000410000042000000'x;</td>
<td>AB (little endian)</td>
</tr>
<tr>
<td>put x $ucs4be.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:
- “$UCS4Xw. Format” on page 279

Informat:
- “$UCS4Xw. Informat” on page 546

$UESCw. Format

Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode escape (UESC) representation.

Category: Character
Alignment: Left
Syntax
$UESCw.

Syntax Description

$w$

specifies the width of the input field.

Default 8

Range 1–32000

Details

If the characters are not available on all operating environments, for example, 0–9, a–z, A–Z, they must be represented in UESC. $UESCw. can be nested.

Comparisons

The $UESCw. format performs processing that is opposite of the $UESCEw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='¥' ;</td>
<td>¥u5927</td>
</tr>
<tr>
<td>y='u5927'</td>
<td>¥uu5927</td>
</tr>
<tr>
<td>z='uu5927';</td>
<td>¥uuu5927</td>
</tr>
<tr>
<td>put x = $uesc10. ;</td>
<td></td>
</tr>
<tr>
<td>put y = $uesc10. ;</td>
<td></td>
</tr>
<tr>
<td>put z = $uesc10. ;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

• “$UESCEw. Format” on page 283

Informats:

• “$UESCw. Informat” on page 549
• “$UESCEw. Informat” on page 550
$UESCEw. Format

Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session.

**Category:** Character  
**Alignment:** Left

### Syntax

$UESCEw$

### Syntax Description

`w` specifies the width of the output field.

- **Default:** 8
- **Range:** 1–32000

### Details

If the data is not supported by the encoding of the current SAS session, the data remains in UESC.

### Comparisons

The $UESCEw$ format performs processing that is the opposite of the $UESCw$ format.

### Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put('¥u5927', $uesce10.);</td>
<td>x=¥u5927</td>
</tr>
<tr>
<td>x=put('¥uu5927', $uesce10.);</td>
<td>x=¥u5927</td>
</tr>
<tr>
<td>x=put('¥uuu5927', $uesce10.);</td>
<td>x=¥u5927</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “$UESCw. Format” on page 281
$UNCRw. Format

Processes a character string that is encoded in the current SAS session, and then writes the character string in numeric character representation (NCR).

**Category:** Character

**Alignment:** Left

### Syntax

$UNCRw.$

### Syntax Description

$w$

- Specifies the width of the output field.

**Default:** 8

**Range:** 1–32000

### Comparisons

The $UNCRw.$ format performs processing that is the opposite of the $UNCREw.$ format.

### Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='91E5'x ; /* Japanese ' 한 ' in Shift-JIS */</td>
<td>9122823</td>
</tr>
<tr>
<td>y='abc';</td>
<td>abc</td>
</tr>
<tr>
<td>put x $uncr10.;</td>
<td></td>
</tr>
<tr>
<td>put y $uncr10.;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Formats:**
$UNCREw. Format

Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session.

**Category:** Character

**Alignment:** Left

### Syntax

\$UNCRE\_w.

#### Syntax Description

\w

- specifies the width of the output field.

- **Default**: 8

- **Range**: 1–32000

### Details

National characters should be represented in NCR.

### Comparisons

The \$UNCREw. format performs processing that is the opposite of the \$UNCRw. format.

### Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='&amp;2823;abc';</td>
<td>&amp;abc</td>
</tr>
<tr>
<td>put x $uncr10.;</td>
<td></td>
</tr>
</tbody>
</table>

---
$UPARENw. Format

Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation.

**Category:** Character

**Alignment:** Left

### Syntax

$UPARENW.

### Syntax Description

\( w \)

specifies the width of the output field.

- **Default:** 8
- **Range:** 27–32000

### Details

The character string is encoded with parentheses and Unicode hexadecimal representation.

### Comparisons

The $UPARENW. format performs processing that is the opposite of the $UPARENEW. format.

### Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| ---+-------+---+---+---+---+
| 1   | 2      | 3  |

---

See Also

**Formats:**

- “$UNCRw. Format” on page 284

**Informs:**

- “$UNCRw. Informat” on page 551
- “$UNCREw. Informat” on page 552
$UPARENEw. Format

Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session.

**Category:** Character

**Alignment:** Left

### Syntax

$UPARENEw$

### Syntax Description

**w**

specifies the width of the output field.

- **Default:** 8
- **Range:** 1–32000

### Comparisons

The $UPARENEw$ format performs processing that is the opposite of the $SUPARENw$ format.

### Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.
Statements | Results
-----+----

```
x='<u0061><u0062><u0063><u0033>';
put x $uparene4.; 
```

See Also

Formats:
- “$UPARENw. Format” on page 286

Informats:
- “$UPARENw. Informat” on page 553
- “$UPARENw. Informat” on page 554

$UTF8Xw. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in universal transformation format (UTF-8) encoding.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

```
$UTF8Xw:
```

Syntax Description

```
w
```

specifies the width of the output field. Specify enough width to include all of the characters in the variable. The width of the characters is dependent on the code point value of the individual characters.

| Default | 8 |
| Range   | 2–32767 |

Comparisons

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

Statements | Results
-----+----

```
-----+----
Statements | Results
--- | ---
x = '91E5'x; /* Japanese 'あ' in Shift-JIS */ x='e5a4a7'x
put x $utf8x10.;

See Also

Formats:
- “$UCS2Bw. Format” on page 266
- “$UCS2Lw. Format” on page 269
- “$UCS2Xw. Format” on page 271

Informats:
- “$UCS2Bw. Informat” on page 538
- “$UCS2Lw. Informat” on page 540
- “$UCS2Xw. Informat” on page 542

$UTF8XEw. Format

Processes a character string that is in universal transformation format (UTF-8), and then writes the character string in the encoding of the current SAS session.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

$UTF8XEw.

Syntax Description

`w`

specifies the width of the output field. Specify enough width to include all of the characters in the variable. The width of the characters is dependent on the code point value of the individual characters.

| Default | 8 |
| Range | 1-32000 |

Comparisons

The $UTF8XEw. format performs processing that is the opposite to the $UTF8Xw. format.
Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = unicode('u5927'); put x $utf8xe10.;</td>
<td>🗣</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UTF8Xw. Format” on page 288

Informats:
- “$UTF8Xw. Informat” on page 557

$VSLOGw. Format

Processes a character string that is in visual order, and then writes the character string in left-to-right logical order.

**Category:** BIDI Text Handling

**Alignment:** Left

**Syntax**

$VSLOGw.

**Syntax Description**

`w` specifies the width of the output field.

- **Default:** 200
- **Range:** 1–32767

**Details**

The $VSLOGw. format is used when transferring data that is stored in visual order. An example is transferring data from a UNIX server to a Windows client.

**Note:** The $VSLOGw. format does not correctly process all combinations of data strings.
Comparisons

The $VSLOGw$ format performs processing that is opposite to the $VSLOGRw$ format.

Example

The following example uses the Hebrew input value of “ﬂight”.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $vslog12.;</td>
<td>ﬂight</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “computer”.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $vslog12.;</td>
<td>ناجم computer</td>
</tr>
</tbody>
</table>

See Also

Format:
- “$VSLOGRw. Format” on page 291

Informats:
- “$VSLOGw. Informat” on page 558
- “$VSLOGRw. Informat” on page 559
**Syntax Description**

\[ w \]

specifies the width of the output field.

---

**Details**

The \$VSLOGRw. format is used when transferring data that is stored in visual order. An example is transferring data from a UNIX server to a Windows client.

*Note:* The \$VSLOGRw. format does not correctly process all combinations of data strings.

**Comparisons**

The \$VSLOGRw. format performs processing that is opposite to the \$VSLOGw. format.

**Example**

The following example uses the Hebrew input value of “ﬂight.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12; flight</td>
<td>נאשכ</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “ذآت” computer.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12; ذآت</td>
<td>computer</td>
</tr>
</tbody>
</table>

**See Also**

**Informats:**

- “\$VSLOGw. Informat” on page 558
- “\$VSLOGRw. Informat” on page 559
**WEEKUw. Format**

Writes a week number in decimal format by using the U algorithm.

- **Category:** Date and Time
- **Alignment:** Left

**Syntax**

`WEEKUw`.  

**Syntax Description**

`w`

- Specifies the width of the output field.

  - **Default:** 11
  - **Range:** 3–200

**Details**

The `WEEKUw` format writes a week-number format. The `WEEKUw` format writes the various formats depending on the specified width. Algorithm U calculates the SAS date value by using the number of the week within the year (Sunday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td><code>Www</code></td>
<td><code>w01</code></td>
</tr>
<tr>
<td>5-6</td>
<td><code>yyWww</code></td>
<td><code>03W01</code></td>
</tr>
<tr>
<td>7-8</td>
<td><code>yyWwwd</code></td>
<td><code>03W0101</code></td>
</tr>
<tr>
<td>9-10</td>
<td><code>yyyyWwwd</code></td>
<td><code>2003W0101</code></td>
</tr>
<tr>
<td>11-200</td>
<td><code>yyyy-Www-dd</code></td>
<td><code>2003-W01-01</code></td>
</tr>
</tbody>
</table>

**Comparisons**

The `WEEKVw` format writes the week number as a decimal number in the range 01–53, with weeks that begin on a Monday and week 1 of the year including both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The `WEEKWw` format writes the week number of the year as a decimal number in the range 00–53, with
Monday as the first day of week 1. The WEEKUw. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

**Example**

```sas
sasdate = '01JAN2003'd;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,weeku3.);</td>
<td>W00</td>
</tr>
<tr>
<td>w=put(sasdate,weeku5.);</td>
<td>03W00</td>
</tr>
<tr>
<td>x=put(sasdate,weeku7.);</td>
<td>03W0004</td>
</tr>
<tr>
<td>y=put(sasdate,weeku9.);</td>
<td>2003W0004</td>
</tr>
<tr>
<td>z=put(sasdate,weeku11.);</td>
<td>2003-W00-04</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

- “WEEKVw. Format” on page 294
- “WEEKWw. Format” on page 296

---

**WEEKVw. Format**

Writes a week number in decimal format by using the V algorithm.

**Category:** Date and Time  
**Alignment:** Left

**Syntax**

```
WEEKVw:
```

**Syntax Description**

```
w
```

specifies the width of the output field.

**Default:** 11  
**Range:** 3–200
Details

The **WEEKVw**. format writes the various formats depending on the specified width. Algorithm V calculates the SAS date value, and the number-of-the-week value is represented as a decimal number in the range 01–53, with a leading zero and maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For example, the fifth week of the year would be represented as 06.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Wwww</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWwww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yyWwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWwwdd</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>

Comparisons

The **WEEKVw**. format writes the week number as a decimal number in the range 01–53, with weeks that begin on a Monday and week 1 of the year including both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The **WEEKWw**. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The **WEEKUw**. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Example

```sas
sasdate='01JAN2003'd;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate, weekv3.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate, weekv5.);</td>
<td>03W01</td>
</tr>
<tr>
<td>x=put(sasdate, weekv7.);</td>
<td>03W0103</td>
</tr>
<tr>
<td>y=put(sasdate, weekv9.);</td>
<td>2003W0103</td>
</tr>
<tr>
<td>z=put(sasdate, weekv11.);</td>
<td>2003-W01-03</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:
- “WEEKUw. Format” on page 293
- “WEEKWw. Format” on page 296

WEEKWw. Format

WEEKWw. Format writes a week number in decimal format by using the W algorithm.

Category: Date and Time
Alignment: Left

Syntax

\texttt{WEEKWw.}

Syntax Description

\textit{w}

\begin{itemize}
\item \text{specifies the width of the output field.}
\end{itemize}

\begin{tabular}{|c|c|c|}
\hline
\text{Default} & 11 \\
\text{Range} & 3–200 \\
\hline
\end{tabular}

Details

The WEEKWw. format writes the various formats depending on the specified width. Algorithm W calculates the SAS date value using the number of the week within the year (Monday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

For more information about widths, formats, and examples see the following table:

\begin{tabular}{|c|c|c|}
\hline
\text{Widths} & \text{Formats} & \text{Examples} \\
\hline
3-4 & \text{Www} & w01 \\
\hline
5-6 & \text{yyWww} & 03W01 \\
\hline
7-8 & \text{yyWwwdd} & 03W0101 \\
\hline
9-10 & \text{yyyyWwwdd} & 2003W0101 \\
\hline
11-200 & \text{yyyy-Www-dd} & 2003-W01-01 \\
\hline
\end{tabular}
Comparisons

The WEEKVw. format writes the week number as a decimal number in the range 01–53. Weeks beginning on a Monday and on week 1 of the year include both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKWw. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The WEEKUw. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Example

sasdate = '01JAN2003'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=put(sasdate,weekw3.);</td>
<td>W03</td>
</tr>
<tr>
<td>w=put(sasdate,weekw5.);</td>
<td>03W03</td>
</tr>
<tr>
<td>x=put(sasdate,weekw7.);</td>
<td>03W0003</td>
</tr>
<tr>
<td>y=put(sasdate,weekw9.);</td>
<td>2003W0003</td>
</tr>
<tr>
<td>z=put(sasdate,weekw11.);</td>
<td>2003-W00-03</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “WEEKUw. Format” on page 293
- “WEEKVw. Format” on page 294

YENw.d Format

Writes numeric values with yen signs, commas, and decimal points.

Category: Numeric
Alignment: Right

Syntax

YENw.d
**Syntax Description**

$w$

specifies the width of the output field.

Default 8

Range 1–32

$d$

specifies the number of digits to the right of the decimal point in the numeric value.

Range 0–9

**Details**

The YEN$w$.d format writes numeric values with a leading yen sign and with a comma that separates every three digits of each value.

The hexadecimal representation of the code for the yen sign character is 5B on EBCDIC systems and 5C on ASCII systems. The monetary character these codes represent might be different in other countries.

**Example**

```plaintext
put cost yen10.2;

data _null_;  
  value=1254.71;  
  put value yen10.2;  
run;
```

<table>
<thead>
<tr>
<th>Cost</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1254.71</td>
<td>¥1,254.71</td>
</tr>
</tbody>
</table>

**See Also**

**Informat:**

- “YENw.d Informat” on page 560

---

**YYWEEKUw. Format**

Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.

**Category:** Date and Time

**Alignment:** Left
Syntax

YYWEEKUw.

Syntax Description

w

specifies the width of the output field.

Default 7

Range 3-8

Details

The YYWEEKUw. format writes a week-number format. The YYWEEKUw. format writes the various formats depending on the specified width. Algorithm U calculates the SAS date value by using the number of the week within the year (Sunday is considered the first day of the week).

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>W01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWww</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-Www</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Comparisons

The YYWEEKUw. format is similar to the WEEKUw. format except that the YYWEEKUw. format does not specify the day-of-week information. Also, the YYWEEKUw. format does not accept any width that is greater than 8.

Example

sasdate = '01JAN2007'd;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>


**Statements**

| u=put(sasdate,yyweeku3.); | $W00$
| v=put(sasdate,yyweeku4.); | $W00$
| w=put(sasdate,yyweeku5.); | 07$W00$
| x=put(sasdate,yyweeku6.); | 07$W00$
| y=put(sasdate,yyweeku7.); | 2007$W00$
| z=put(sasdate,yyweeku8.); | 2007-$W00$
| put u; | 
| put v; | 
| put w; | 
| put x; | 
| put y; | 
| put z; | 

**See Also**

**Format:**

- “WEEKUw. Format” on page 293

---

**YYWEEKVw. Format**

Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.

**Category:** Date and Time

**Alignment:** Left

**Syntax**

**YYWEEKV**

**Syntax Description**

`w`

specifies the width of the output field.

- Default: 7
- Range: 3–8

**Details**

The YYWEEKVw. format writes the various formats depending on the specified width. Algorithm V calculates the SAS date value, and the number-of-the-week value is represented as a decimal number in the range 01–53, with a leading zero and maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For example, the fifth week of the year would be represented as 06.

For more information about widths, formats, and examples see the following table:
Comparisons

The YYWEEEKVw. format is similar to the WEEKVw. format except that the YYWEEEKVw. format does not specify the day-of-week information. Also, the YYWEEEKVw. format does not accept a width that is greater than 8.

Example

```
sasdate = '01JAN2007'd;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweekv3.);</td>
<td>W01</td>
</tr>
<tr>
<td>v=put(sasdate,yyweekv4.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate,yyweekv5.);</td>
<td>07W01</td>
</tr>
<tr>
<td>x=put(sasdate,yyweekv6.);</td>
<td>07W01</td>
</tr>
<tr>
<td>y=put(sasdate,yyweekv7.);</td>
<td>2007W01</td>
</tr>
<tr>
<td>z=put(sasdate,yyweekv8.);</td>
<td>2007-W01</td>
</tr>
<tr>
<td>put u; put v; put w; put x; put y; put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Format:

- “WEEKVw. Format” on page 294
Syntax

YYWEEKWw.

Syntax Description

w

specifies the width of the output field.

Default 7

Range 3–8

Details

The YYWEEKWw. format writes the various formats depending on the specified width. Algorithm W calculates the SAS date value using the number of the week within the year.

For more information about widths, formats, and examples see the following table:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Wwww</td>
<td>W01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyyyWww</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWww</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-Www</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Comparisons

The YYWEEKWw. format is similar to the WEEKWw. format except that the YYWEEKWw. format does not specify the day-of-week information. Also, the YYWEEKWw. format does not accept any width that is greater than 8.

Example

sasdate = '01JAN2007'd

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
|            | --+-----+
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweekw3.);</td>
<td>W01</td>
</tr>
<tr>
<td>v=put(sasdate,yyweekw4.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate,yyweekw5.);</td>
<td>07W01</td>
</tr>
<tr>
<td>x=put(sasdate,yyweekw6.);</td>
<td>07W01</td>
</tr>
<tr>
<td>y=put(sasdate,yyweekw7.);</td>
<td>2007W01</td>
</tr>
<tr>
<td>z=put(sasdate,yyweekw8.);</td>
<td>2007-W01</td>
</tr>
<tr>
<td>put u;</td>
<td></td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “WEEKWw. Format” on page 296
Part 5

Functions for NLS

Chapter 11
  Internationalization Compatibility for SAS String Functions . . . . 307

Chapter 12
  Dictionary of Functions for NLS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 325
Chapter 11
Internationalization Compatibility for SAS String Functions

SAS provides string functions and CALL routines that enable you to easily manipulate your character data. Many of the original SAS string functions assume that the size of one character is always one byte. This process works well for data in a single-byte character set (SBCS). However, when some of these functions and CALL routines are used with data in a double-byte character set (DBCS) or multi-byte character set (MBCS), the data is often handled improperly and produce incorrect results.

DBCS encodings require a varying number of bytes to represent each character. MBCS is sometimes used as a synonym for DBCS.

To solve this problem SAS introduced a set of string functions and CALL routines, called K functions, for those string manipulations where DBCS and MBCS data must be handled carefully. This page shows the level of I18N compatibility for each SAS string function. I18N is the abbreviation for internationalization. Compatibility indicates whether a program using a particular string function can be adapted to different languages and locales without program changes.

The user needs to understand the difference between byte-based offset-length and character-based offset-length in order to use the K functions properly. Most K functions require the character-based offset or length. Under SBCS environments, the byte-based unit is identical to character-based unit. However, under DBCS or MBCS environment, there are significant differences, and programmers need to distinguish them. The users might need to change the programming logic in order to use the K functions. Most K functions require strings encoded in current SAS session encoding.

String functions are assigned I18N levels depending on whether the functions can process DBCS, MBCS, or SBCS. Here are descriptions of the levels:

I18N Level 0
This function is designed for SBCS data. Do not use this function to process DBCS or MBCS data.

I18N Level 1
This function should be avoided, if possible, if you are processing DBCS or MBCS data. The I18N Level 1 functions might not work correctly with DBCS or MBCS encodings under certain circumstances.
I18N Level 2

This function can be used for SBCS, DBCS, and MBCS (UTF-8) data.

**Table 11.1  SAS String Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>I18N Level 0</th>
<th>I18N Level 1</th>
<th>I18N Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ANORM420 Function” (p. 330)</td>
<td>Returns a normalized string from an input string encoded in EBCDIC420.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYALNUM Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for an alphanumeric character, and returns the first position at which the character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYALPHA Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for an alphabetic character, and returns the first position at which the character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYCNTRL Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a control character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYDIGIT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a digit, and returns the first position at which the digit is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYFIRST Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is valid as the first character in a SAS variable name under VALIDVARNAME= V7, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYGRAPH Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a graphical character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“ANYLOWER Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a lowercase letter, and returns the first position at which the letter is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYNAME Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is valid in a SAS variable name under VALIDVARNAME=V7, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYPRINT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a printable character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYPUNCT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a punctuation character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYSPACE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a white-space-character (blank, horizontal and vertical tab, carriage return, line feed, and form feed). Returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYUPPER Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for an uppercase letter, and returns the first position at which the letter is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“ANYXDIGIT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a hexadecimal character that represents a digit, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“BASECHAR Function” (p. 334)</td>
<td>Converts characters to base characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“BYTE Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns one character in the ASCII or the EBCDIC collating sequence.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CAT Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Does not remove leading or trailing blanks, and returns a concatenated character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CATQ Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Concatenates character or numeric values by using a delimiter to separate items and by adding quotation marks to strings that contain the delimiter.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CATS Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Removes leading and trailing blanks, and returns a concatenated character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CATT Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Removes trailing blanks, and returns a concatenated character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CATX Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Removes leading and trailing blanks, inserts delimiters, and returns a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CHAR Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a single character from a specified position in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CHOOSEC Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a character value that represents the results of choosing from a list of arguments.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“CHOOSEN Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a numeric value that represents the results of choosing from a list of arguments.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“COALESCE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the first nonmissing value from a list of numeric arguments.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COLLATE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character string in ASCII or EBCDIC collating sequence.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPARE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the position of the leftmost character by which two strings differ, or returns 0 if there is no difference.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPBL Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes multiple blanks from a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPGED Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the generalized edit distance between two strings.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPLEV Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the Levenshtein edit distance between two strings.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COMPRESSION Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character string with specified characters removed from the original string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COUNT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Counts the number of times that a specified substring appears within a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COUNTC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Counts the number of characters in a string that appear or do not appear in a list of characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“COUNTW Function” in SAS Functions and CALL Routines: Reference</td>
<td>Counts the number of words in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“DEQUOTE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes matching quotation marks from a character string that begins with a quotation mark, and deletes all characters to the right of the closing quotation mark.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“FIND Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches for a specific substring of characters within a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“FINDC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a string for any character in a list of characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“FINDW Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the character position of a word in a string, or returns the number of the word in a string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“FIRST Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the first character in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“GETLOCENV Function” (p. 338)</td>
<td>Returns the current locale/language environment.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLANGUAGE Function” (p. 339)</td>
<td>Returns the current two-letter language code.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXLOCALE Function” (p. 340)</td>
<td>Returns the POSIX locale value for a SAS locale.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“GETPXREGION Function” (p. 341)</td>
<td>Returns the current two-letter region code.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“HTMLDECODE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Decodes a string that contains HTML numeric character references or HTML character entity references, and returns the decoded string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“HTMLENCODE Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Encodes characters using HTML character entity references, and returns the encoded string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“IFC Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a character value based on whether an expression is true, false, or missing.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“IFN Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a numeric value based on whether an expression is true, false, or missing.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“INDEX Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character expression for a string of characters, and returns the position of the string's first character for the first occurrence of the string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“INDEXC Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character expression for any of the specified characters, and returns the position of that character.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“INDEXW Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character expression for a string that is specified as a word, and returns the position of the first character in the word.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOMPARE Function” (p. 342)</td>
<td>Returns the result of a comparison of character expressions.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOMPRESS Function” (p. 343)</td>
<td>Removes specified characters from a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOUNT Function” (p. 344)</td>
<td>Returns the number of double-byte characters in an expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KCVT Function” (p. 345)</td>
<td>Converts data from one type of encoding data to another encoding data.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEX Function” (p. 347)</td>
<td>Searches a character expression for a string of characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEXB Function” (p. 348)</td>
<td>Searches a character expression for a string of characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEXCB Function” (p. 349)</td>
<td>Searches a character expression for specified characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KINDEXC Function” (p. 350)</td>
<td>Searches a character expression for specified characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLEFT Function” (p. 351)</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO-SI.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLENGTH Function” (p. 352)</td>
<td>Returns the length of an argument.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KLOWCASE Function” (p. 353)</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPCASE Function” (p. 354)</td>
<td>Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPCCHAR Function” (p. 357)</td>
<td>Converts special characters to normal characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KPROPDATA Function” (p. 358)</td>
<td>Removes or converts unprintable characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KREVERSE Function” (p. 360)</td>
<td>Reverses a character expression.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KRIGHT Function” (p. 361)</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO-SI.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KSCAN Function” (p. 362)</td>
<td>Selects a specified word from a character expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KSTRCAT Function” (p. 363)</td>
<td>Concatenates two or more character expressions.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KSTRIP Function” (p. 364)</td>
<td>Removes leading and trailing blanks from a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KSUBSTR Function” (p. 366)</td>
<td>Extracts a substring from an argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KSUBSTRB Function” (p. 367)</td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KTRANSLATE Function” (p. 368)</td>
<td>Replaces specific characters in a character expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KTRIM Function” (p. 370)</td>
<td>Removes trailing DBCS blanks and SO-SI from character expressions.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KTRUNCATE Function” (p. 371)</td>
<td>Truncates a character string to a specified length in byte units without breaking multibyte characters.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KUPCASE Function” (p. 372)</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KUPDATE Function” (p. 373)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KUPDATEB Function” (p. 375)</td>
<td>Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KUPDATES Function” (p. 376)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KVERIFYB Function” (p. 379)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KVERIFY Function” (p. 378)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“LEFT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Left-aligns a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“LENGTH Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the length of a non-blank character string, excluding trailing blanks, and returns 1 for a blank character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“LENGTHC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, including trailing blanks.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“LENGTHM Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the amount of memory (in bytes) that is allocated for a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“LENGTHN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the length of a character string, excluding trailing blanks.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“LOWCASE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“MD5 Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the result of the message digest of a specified string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“MISSING Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a numeric result that indicates whether the argument contains a missing value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“MVALID Function” in SAS Functions and CALL Routines: Reference</td>
<td>Checks the validity of a character string for use as a SAS member name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“NLDATE Function” (p. 380)</td>
<td>Converts the SAS date value to the date value of the specified locale by using the date format descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLDATM Function” (p. 383)</td>
<td>Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLTIME Function” (p. 386)</td>
<td>Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NLITERAL Function” in SAS Functions and CALL Routines: Reference</td>
<td>Converts a character string that you specify to a SAS name literal.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTALNUM Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a non-alphanumeric character, and returns the first position at which the character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTALPHA Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a nonalphabetic character, and returns the first position at which the character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTCNTRL Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a control character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“NOTDIGIT Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for any character that is not a digit, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTFIRST Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for an invalid first character in a SAS variable name under VALIDVARNAME= V7, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTGRAPH Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a non-graphical character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTLOWER Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a character that is not a lowercase letter, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTNAME Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for an invalid character in a SAS variable name under VALIDVARNAME= V7, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTPRINT Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Searches a character string for a nonprintable character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“NOTPUNCT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a punctuation character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTSPACE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a whitespace-character (blank, horizontal and vertical tab, carriage return, line feed, and form feed), and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTUPPER Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not an uppercase letter, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NOTXDIGIT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches a character string for a character that is not a hexadecimal character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“NVALID Function” in SAS Functions and CALL Routines: Reference</td>
<td>Checks the validity of a character string for use as a SAS variable name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PROPCASE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Converts all words in an argument to proper case.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PRXCHANGE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Performs a pattern-matching replacement.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PRXMATCH Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches for a pattern match and returns the position at which the pattern is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“PRXPAREN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the last bracket match for which there is a match in a pattern.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PRXPARSE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Compiles a Perl regular expression (PRX) that can be used for pattern matching of a character value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PRXPOSN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character string that contains the value for a capture buffer.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PUT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a value using a specified format.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PUTC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Enables you to specify a character format at run time.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“PUTN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Enables you to specify a numeric format at run time.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“QUOTE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Adds double quotation marks to a character value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“RANK Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the position of a character in the ASCII or EBCDIC collating sequence.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“REPEAT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a character value that consists of the first argument repeated n+1 times.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“REVERSE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Reverses a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“RIGHT Function” in SAS Functions and CALL Routines: Reference</td>
<td>Right-aligns a character expression.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“SASMSG Function” (p. 387)</td>
<td>Specifies a message from a data set. The returned message is based on the current locale and a specified key.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“SASMSGL Function” (p. 390)</td>
<td>Specifies a message from a data set. The message is based on a specified locale value and a specified key value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SCAN Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns the nth word from a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SETLOCALE Function” (p. 395)</td>
<td>Specifies the locale keys for the current SAS locale.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SHA256 Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns the result of the message digest of a specified string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SORTKEY Function” (p. 392)</td>
<td>Creates a linguistic sort key.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“SOUNDEX Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Encodes a string to facilitate searching.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SPEDIS Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“STRIP Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a character string with all leading and trailing blanks removed.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SUBPAD Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a substring that has a length that you specify, using blank padding if necessary.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SUBSTR (left of =) Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Extracts a substring from an argument.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SUBSTR (right of =) Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Extracts a substring from an argument.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SUBSTRN Function” in <em>SAS Functions and CALL Routines: Reference</em></td>
<td>Returns a substring, allowing a result with a length of zero.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“TRANSLATE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Replaces specific characters in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRANSTRN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Replaces or removes all occurrences of a substring in a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TRANTAB Function” on page 412</td>
<td>Transcodes data by using the specified translation table.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRANWRD Function” in SAS Functions and CALL Routines: Reference</td>
<td>Replaces or removes all occurrences of a substring in a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TRIM Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes trailing blanks from a character string, and returns one blank if the string is missing.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TRIMN Function” in SAS Functions and CALL Routines: Reference</td>
<td>Removes trailing blanks from character expressions, and returns a string with a length of zero if the expression is missing.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“TZONEDSTNAME Function” (p. 407)</td>
<td>Returns a daylight savings time name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONEDSTOFF Function” (p. 408)</td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“TZONEID Function” (p. 402)</td>
<td>Returns the current time zone ID.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“TZONENAME Function” (p. 404)</td>
<td>Returns the current standard or daylight savings time, time zone name.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“ZONEFF Function” (p. 405)</td>
<td>Returns the user time zone offset.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONES2U Function” (p. 406)</td>
<td>Converts a SAS date time value to a UTC date time value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“TZONESTTNAME Function” (p. 409)</td>
<td>Returns a standard time zone name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“TZONESTTOFF Function” (p. 410)</td>
<td>Returns the time zone offset value for the specified standard time.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TZONEU2S Function” (p. 411)</td>
<td>Converts a UTC date time value to a SAS date time value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODE Function” (p. 413)</td>
<td>Converts Unicode characters to the current SAS session encoding.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UNICODEC Function” (p. 414)</td>
<td>Converts characters in the current SAS session encoding to Unicode characters.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“UNICODELEN Function” (p. 415)</td>
<td>Specifies the length of the character unit for the Unicode data.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UNICODEWIDTH Function” (p. 416)</td>
<td>Specifies the length of a display unit for the Unicode data.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“UPCASE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Converts all letters in an argument to uppercase.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“URLDECODE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a string that was decoded using the URL escape syntax.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“URLENCODE Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns a string that was encoded using the URL escape syntax.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“VERIFY Function” in SAS Functions and CALL Routines: Reference</td>
<td>Returns the position of the first character in a string that is not in any of several other strings.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“WHICHC Function” in SAS Functions and CALL Routines: Reference</td>
<td>Searches for a character value that is equal to the first argument, and returns the index of the first matching value.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 11 • Internationalization Compatibility for SAS String Functions
Chapter 12
Dictionary of Functions for NLS

Functions by Category .......................................................... 326

Dictionary ................................................................. 330
ANORM420 Function .................................................. 330
BASECHAR Function .................................................. 334
ENCODCOMPAT Function ........................................ 336
ENCODISVALID Function ......................................... 337
GETLOCENV Function ............................................. 338
GETPXLANGUAGE Function ..................................... 339
GETPXLOCALE Function .......................................... 340
GETPXREGION Function ........................................ 341
KCOMPARE Function .................................................. 342
KCOMPRESS Function ............................................. 343
KCOUNT Function .................................................... 344
KCOUNT Function .................................................... 344
KCVT Function .......................................................... 345
KINDEX Function ...................................................... 347
KINDEXB Function .................................................... 348
KINDEXCB Function ................................................. 349
KINDEXC Function .................................................... 350
KLEFT Function .......................................................... 351
KLENGTH Function ................................................... 352
KLOWCASE Function .................................................. 353
KPROPCHAR Function ............................................... 354
KPROPCASE Function ............................................... 357
KPROPDATA Function .............................................. 358
KREVERSE Function .................................................. 360
KRIGHT Function ...................................................... 361
KSCAN Function .......................................................... 362
KSTRCAT Function ...................................................... 363
KSTRIP Function .......................................................... 364
KSUBSTR Function ...................................................... 364
KSUBSTRB Function ................................................... 366
KTRANSLATE Function ............................................. 367
KTRIM Function .......................................................... 368
KTRUNCATE Function ............................................... 370
KUPDATE Function ..................................................... 371
KUPDATES Function .................................................... 372
KUPDATEB Function .................................................... 373
KVERIFY Function ....................................................... 375
KVERIFYB Function ..................................................... 376
NLDATE Function ......................................................... 378
## Functions by Category

The following categories relate to NLS issues:

### Table 12.1 Categories of NLS Functions

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>processes character data</td>
</tr>
<tr>
<td>Currency Conversion</td>
<td>converts one currency to another currency</td>
</tr>
<tr>
<td>DBCS</td>
<td>processes double-byte character set.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>processes data and time data.</td>
</tr>
<tr>
<td>Locale</td>
<td>processes data based on the specified locale.</td>
</tr>
<tr>
<td>Variable Information</td>
<td>processes variable information.</td>
</tr>
</tbody>
</table>

### Language Elements

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>ANORM420 Function (p. 330)</td>
<td>Returns a normalized string from an input string encoded in EBCDIC420.</td>
</tr>
<tr>
<td></td>
<td>BASECHAR Function (p. 334)</td>
<td>Converts characters to base characters.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>KCVT Function (p. 345)</strong></td>
<td>Converts data from one type of encoding data to another type of encoding data.</td>
</tr>
<tr>
<td></td>
<td><strong>KSTRIP Function (p. 364)</strong></td>
<td>Removes leading and trailing blanks from a character string.</td>
</tr>
<tr>
<td></td>
<td><strong>TRANTAB Function (p. 412)</strong></td>
<td>Transcodes data by using the specified translation table.</td>
</tr>
<tr>
<td></td>
<td><strong>UNICODE Function (p. 413)</strong></td>
<td>Converts Unicode characters to the current SAS session encoding.</td>
</tr>
<tr>
<td></td>
<td><strong>UNICODEC Function (p. 414)</strong></td>
<td>Converts characters in the current SAS session encoding to Unicode characters.</td>
</tr>
<tr>
<td></td>
<td><strong>UNICODELEN Function (p. 415)</strong></td>
<td>Specifies the length of the character unit for the Unicode data.</td>
</tr>
<tr>
<td></td>
<td><strong>UNICODEWIDTH Function (p. 416)</strong></td>
<td>Specifies the length of a display unit for the Unicode data.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>NLDATE Function (p. 380)</strong></td>
<td>Converts the SAS date value to the date value of the specified locale by using the date format descriptors.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>NLDATM Function (p. 383)</strong></td>
<td>Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>NLTIME Function (p. 386)</strong></td>
<td>Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONEID Function (p. 402)</strong></td>
<td>Returns the current time zone ID.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONENAME Function (p. 404)</strong></td>
<td>Returns the current standard or daylight savings time, time zone name.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONEOFF Function (p. 405)</strong></td>
<td>Returns the user time zone offset.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONES2U Function (p. 406)</strong></td>
<td>Converts a SAS date time value to a UTC date time value.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONEDSTNAME Function (p. 407)</strong></td>
<td>Returns a daylight savings time name.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONEDSTOFF Function (p. 408)</strong></td>
<td>Returns the time zone offset value for the specified daylight savings time.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONESTTNAME Function (p. 409)</strong></td>
<td>Returns a standard time zone name.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONESTTOFF Function (p. 410)</strong></td>
<td>Returns the time zone offset value for the specified standard time.</td>
</tr>
<tr>
<td>Date and Time</td>
<td><strong>TZONEU2S Function (p. 411)</strong></td>
<td>Converts a UTC date time value to a SAS date time value.</td>
</tr>
<tr>
<td>DBCS</td>
<td><strong>KCOMPARE Function (p. 342)</strong></td>
<td>Returns the result of a comparison of character expressions.</td>
</tr>
</tbody>
</table>

*Note: The page number in parentheses indicates the section in the SAS documentation where each function is described.*
<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCOMPRESS Function (p. 343)</td>
<td></td>
<td>Removes specified characters from a character expression.</td>
</tr>
<tr>
<td>KCOUNT Function (p. 344)</td>
<td></td>
<td>Returns the number of double-byte characters in an expression.</td>
</tr>
<tr>
<td>KINDEX Function (p. 347)</td>
<td></td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td>KINDEXB Function (p. 348)</td>
<td></td>
<td>Searches a character expression for a string of characters.</td>
</tr>
<tr>
<td>KINDEXCB Function (p. 349)</td>
<td></td>
<td>Searches a character expression for specified characters.</td>
</tr>
<tr>
<td>KINDEXC Function (p. 350)</td>
<td></td>
<td>Searches a character expression for specified characters.</td>
</tr>
<tr>
<td>KLEFT Function (p. 351)</td>
<td></td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td>KLENGTH Function (p. 352)</td>
<td></td>
<td>Returns the length of an argument.</td>
</tr>
<tr>
<td>KLOWCASE Function (p. 353)</td>
<td></td>
<td>Converts all letters in an argument to lowercase.</td>
</tr>
<tr>
<td>KPROPCHAR Function (p. 357)</td>
<td></td>
<td>Converts special characters to normal characters.</td>
</tr>
<tr>
<td>KPROPDATA Function (p. 358)</td>
<td></td>
<td>Removes or converts unprintable characters.</td>
</tr>
<tr>
<td>KREVERSE Function (p. 360)</td>
<td></td>
<td>Reverses a character expression.</td>
</tr>
<tr>
<td>KRIGHT Function (p. 361)</td>
<td></td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.</td>
</tr>
<tr>
<td>KSCAN Function (p. 362)</td>
<td></td>
<td>Selects a specified word from a character expression.</td>
</tr>
<tr>
<td>KSTRCAT Function (p. 363)</td>
<td></td>
<td>Concatenates two or more character expressions.</td>
</tr>
<tr>
<td>KSUBSTR Function (p. 366)</td>
<td></td>
<td>Extracts a substring from an argument.</td>
</tr>
<tr>
<td>KSUBSTRB Function (p. 367)</td>
<td></td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
</tr>
<tr>
<td>KTRANSLATE Function (p. 368)</td>
<td></td>
<td>Replaces specific characters in a character expression.</td>
</tr>
<tr>
<td>KTRIM Function (p. 370)</td>
<td></td>
<td>Removes trailing DBCS blanks and SO/SI from character expressions.</td>
</tr>
<tr>
<td>KTRUNCATE Function (p. 371)</td>
<td></td>
<td>Truncates a string to a specified length in byte unit without breaking multibyte characters.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Language Elements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KUPCASE Function (p. 372)</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
</tr>
<tr>
<td>KUPDATE Function (p. 373)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
</tr>
<tr>
<td>KUPDATEB Function (p. 375)</td>
<td>Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.</td>
<td></td>
</tr>
<tr>
<td>KUPDATES Function (p. 376)</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
</tr>
<tr>
<td>KVERIFY Function (p. 378)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
</tr>
<tr>
<td>KVERIFYB Function (p. 379)</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td></td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENCODCOMPAT Function (p. 336)</td>
<td>Verifies the transcoding compatibility between two encodings.</td>
<td></td>
</tr>
<tr>
<td>ENCODISVALID Function (p. 337)</td>
<td>Verifies a valid encoding name.</td>
<td></td>
</tr>
<tr>
<td><strong>Locale</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETLOCENV Function (p. 338)</td>
<td>Returns the current locale/language environment.</td>
<td></td>
</tr>
<tr>
<td>GETPXLANGUAGE Function (p. 339)</td>
<td>Returns the current two-letter language code.</td>
<td></td>
</tr>
<tr>
<td>GETPXLOCALE Function (p. 340)</td>
<td>Returns the POSIX locale value for a SAS locale.</td>
<td></td>
</tr>
<tr>
<td>GETPXREGION Function (p. 341)</td>
<td>Returns the current two-letter region code.</td>
<td></td>
</tr>
<tr>
<td>SASMSG Function (p. 387)</td>
<td>Specifies a message from a data set. The returned message is based on the current locale and a specified key.</td>
<td></td>
</tr>
<tr>
<td>SASMSGL Function (p. 390)</td>
<td>Specifies a message from a data set. The message is based on a specified locale value and a specified key value.</td>
<td></td>
</tr>
<tr>
<td>SORTKEY Function (p. 392)</td>
<td>Creates a linguistic sort key.</td>
<td></td>
</tr>
<tr>
<td>SETLOCALE Function (p. 395)</td>
<td>Specifies the locale keys for the current SAS locale.</td>
<td></td>
</tr>
<tr>
<td><strong>Variable Information</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARTRANSCODE Function (p. 417)</td>
<td>Returns the transcode attribute of a SAS data set variable.</td>
<td></td>
</tr>
<tr>
<td>VTRANSCODE Function (p. 419)</td>
<td>Returns a value that indicates whether transcoding is enabled for the specified character variable.</td>
<td></td>
</tr>
<tr>
<td>VTRANSCODEX Function (p. 420)</td>
<td>Returns a value that indicates whether transcoding is enabled for the specified argument.</td>
<td></td>
</tr>
</tbody>
</table>
ANORM420 Function

Returns a normalized string from an input string encoded in EBCDIC420.

**Category:** Character

**Restriction:** This function is assigned an I18N Level 1 status. If possible, avoid I18N Level 1 functions if you are using a non-English language. Under certain circumstances, the I18N Level 1 functions might not work correctly with Double-Byte Character Set (DBCS) or Multi-Byte Character Set (MBCS) encodings. For more information, see Internationalization Compatibility on page 307.

**Syntax**

ANORM420 (string,<modifiers>)

**Required Argument**

*string*

specifies an input string that is encoded in EBCDIC420.

**Optional Argument**

*modifiers*

are character constants, variables, or expressions that specify one or more modifiers. The following modifiers can be in uppercase or lowercase:

- g ignores the ligature
- i ignores the Arabic-Indic numbers
- p ignores mapping of shaped characters
- s ignores the addition of space
- t ignores transcoding
- z ignores mapping of the zero-length-space character

**Details**

The ANORM420 function processes data that is encoded in EBCDIC420. Refer to the following tables for information about the modifier variables:
The ANORM420 function replaces the following character and code points with the appropriate ligature unless the modifier \textit{g} is specified.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Character & Unicode Value & From & To & Description \\
\hline
\begin{center}ﻵ\textsubscript{U+FEF5}\end{center} & 0xB2 & \textsubscript{0xB1 + 0x47} & Arabic Ligature Lam with Aleph with Madda above - Isolated form \\
\begin{center}ﻶ\textsubscript{U+FEF6}\end{center} & 0xB3 & \textsubscript{0xB1 + 0x47} & Arabic Ligature Lam with Aleph with Madda above - Final form \\
\begin{center}ﻷ\textsubscript{U+FEF7}\end{center} & 0xB4 & \textsubscript{0xB1 + 0x49} & Arabic Ligature Lam with Aleph with Hamza above - Isolated form \\
\begin{center}ﻹ\textsubscript{U+FEF8}\end{center} & 0xB5 & \textsubscript{0xB1 + 0x49} & Arabic Ligature Lam with Aleph with Hamza above - Final form \\
\begin{center}ﻻ\textsubscript{U+FEFC}\end{center} & 0xB8 & \textsubscript{0xB1 + 0x56} & Arabic Ligature Lam with Aleph - Isolated form \\
\begin{center}ﻼ\textsubscript{U+FEFD}\end{center} & 0xB9 & \textsubscript{0xB1 + 0x56} & Arabic Ligature Lam with Aleph - Final form \\
\hline
\end{tabular}
\caption{Modifier \textit{g}: Arabic Ligatures}
\end{table}

The ANORM420 function remaps the Arabic-Indic numbers to the digit unless the modifier \textit{i} is specified and no transcoding occurs.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
From & \begin{center}٠\textsubscript{U+0660}\end{center} & \begin{center}١\textsubscript{U+0661}\end{center} & \begin{center}٢\textsubscript{U+0662}\end{center} & \begin{center}٣\textsubscript{U+0663}\end{center} & \begin{center}٤\textsubscript{U+0664}\end{center} & \begin{center}٥\textsubscript{U+0665}\end{center} & \begin{center}٦\textsubscript{U+0666}\end{center} & \begin{center}٧\textsubscript{U+0667}\end{center} & \begin{center}٨\textsubscript{U+0668}\end{center} & \begin{center}٩\textsubscript{U+0669}\end{center} \\
\hline
Hexadecimal & 0xDF & 0xEA & 0xEB & 0xED & 0xEE & 0xFB & 0xFC & 0xFD & 0xFE \\
To & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
Hexadecimal & 0xF0 & 0xF1 & 0xF2 & 0xF3 & 0xF4 & 0xF5 & 0xF6 & 0xF7 & 0xF8 & 0xF9 \\
\hline
\end{tabular}
\caption{Modifier \textit{i}: Arabic-Indic Numbers}
\end{table}

The ANORM420 function remaps the shaped characters to their unshaped equivalent unless the modifier \textit{p} is specified.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
From & To & \\
\hline
0x43 & U+FE7D & 0x42 & U+0651 \\
0x48 & FE82 & 0x47 & U+0622 \\
0x51 & U+FE84 & 0x49 & U+0623 \\
\hline
\end{tabular}
\caption{Modifier \textit{p}: Shaped Characters Mapping}
\end{table}
<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x57</td>
<td>U+FE8E</td>
<td>0x56</td>
</tr>
<tr>
<td>0x59</td>
<td>U+FE91</td>
<td>0x58</td>
</tr>
<tr>
<td>0x64</td>
<td>U+FE97</td>
<td>0x63</td>
</tr>
<tr>
<td>0x66</td>
<td>U+FE9B</td>
<td>0x65</td>
</tr>
<tr>
<td>0x68</td>
<td>U+FE9F</td>
<td>0x67</td>
</tr>
<tr>
<td>0x70</td>
<td>U+FEA3</td>
<td>0x69</td>
</tr>
<tr>
<td>0x72</td>
<td>U+FEA7</td>
<td>0x71</td>
</tr>
<tr>
<td>0x78</td>
<td>U+FEB3</td>
<td>0x77</td>
</tr>
<tr>
<td>0x8A</td>
<td>U+FEB7</td>
<td>0x80</td>
</tr>
<tr>
<td>0x8C</td>
<td>U+FEBB</td>
<td>0x8B</td>
</tr>
<tr>
<td>0x8E</td>
<td>U+FEBF</td>
<td>0x8D</td>
</tr>
<tr>
<td>0x9B</td>
<td>U+FECA</td>
<td>0x9A</td>
</tr>
<tr>
<td>0x9C</td>
<td>U+FECB</td>
<td>0x9A</td>
</tr>
<tr>
<td>0x9D</td>
<td>U+FECC</td>
<td>0x9A</td>
</tr>
<tr>
<td>0x9F</td>
<td>U+FECE</td>
<td>0x9E</td>
</tr>
<tr>
<td>0xA0</td>
<td>U+FECF</td>
<td>0x9E</td>
</tr>
<tr>
<td>0xAA</td>
<td>U+FED0</td>
<td>0x9E</td>
</tr>
<tr>
<td>0xAC</td>
<td>U+FED3</td>
<td>0xAB</td>
</tr>
<tr>
<td>0xAE</td>
<td>U+FED7</td>
<td>0xAD</td>
</tr>
<tr>
<td>0xB0</td>
<td>U+FEDB</td>
<td>0xAF</td>
</tr>
<tr>
<td>0xBA</td>
<td>U+FEDF</td>
<td>0xB1</td>
</tr>
<tr>
<td>0xBC</td>
<td>U+FEE3</td>
<td>0xBB</td>
</tr>
<tr>
<td>0xBE</td>
<td>U+FEE7</td>
<td>0xBD</td>
</tr>
<tr>
<td>0xCB</td>
<td>U+FEEB</td>
<td>0xBF</td>
</tr>
<tr>
<td>0xCD</td>
<td>U+FEEC</td>
<td>0xBF</td>
</tr>
</tbody>
</table>
The ANORM420 function adds a space after the following characters unless the modifier $s$ is specified.

Table 12.5  Modifier $s$: ignores the addition of a space

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Arabic Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xDB</td>
<td>U+FEF0</td>
<td>U+061F</td>
</tr>
<tr>
<td>0xDD</td>
<td>U+FEF2</td>
<td>U+064A</td>
</tr>
<tr>
<td>0xDE</td>
<td>U+FEF3</td>
<td>U+064A</td>
</tr>
</tbody>
</table>

| ب     | 0x58 (U+0628)    | BEH           |
| ء     | 0x62 (U+0629)    | MARBUTA       |
| ت     | 0x63 (U+062A)    | TEH           |
| ث     | 0x65 (U+062B)    | THEH          |
| ج     | 0x67 (U+062C)    | JEEM          |
| ح     | 0x69 (U+062D)    | HAH           |
| خ     | 0x71 (U+062E)    | KHAH          |
| س     | 0x77 (U+0633)    | SEEN          |
| ش     | 0x80 (U+0634)    | SHEEN         |
| ص     | 0x8B (U+0635)    | SAD           |
| ض     | 0x8D (U+0636)    | DAD           |
| ع     | 0x9A (U+0639)    | AIN           |
| غ     | 0x9B (U+FECA)    | AIN final form|
| ء     | 0x9E (U+063A)    | GHAIN         |
| ء     | 0x9F (U+FECE)    | GHAIN final form|
| ف     | 0xAB (U+0641)    | FEH           |
| ق     | 0xAD (U+0642)    | QAF           |
The ANORM420 function transcodes the normalized string to the session encoding unless the modifier $t$ is specified.

The ANORM420 function remaps the zero-length-break-space character (U+200B), also abbreviated ZWSP, to a space character unless the modifier $z$ is specified. The zero-length character is located at code position 0x45.

**Example**

The following example demonstrates the ANORM420 function:

```plaintext
statements | results
-------------|---------------------------
data _null_; a = '59CD7BC577745'x ; s1 = anorm420(a) ; /* Turn off addition of space and mapping of Arabic-Indic numbers */ s2 = anorm420(a,"s1") ; /* Turn off transcoding */ s3 = anorm420(a,’t’); put s1=$hex20. / s2=$hex20. / s3=$hex20. ; run;
```

**BASECHAR Function**

Converts characters to base characters.
Category: Character
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Syntax

\[
\text{STR=BASECHAR}(\text{<instr>}, (<\text{Unicode type}>))
\]

Required Arguments

- \textit{str}: data string that is converted.
- \textit{instr}: input data string.
- \textit{Unicode type}: If one of the following Unicode character formats is specified, national characters are represented in the specified format:
  - \texttt{ESC}: Unicode Escape (u00cd).
  - \texttt{NCR}: Numeric Character Representation (&#x00c5;).
  - \texttt{PAREN}: Unicode Parenthesis Escape (<u00c5>).

Details

The BASECHAR function reads characters and converts them to base characters. Some characters consist of a base character and one or more accents or combining characters. The BASECHAR function reads the characters in the string and converts them to the base character form without the accents based on Unicode specifications. When Unicode type is specified, national characters are represented in the specified format. For example, the data string Mühlenfließ is converted to Muhlenfließ.

Example

The following examples demonstrate using the Unicode types:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: The SAS session encoding is wlatin1. data cities; length name$24 name_ascii$24; input name; name_ascii = basechar(name); put name_ascii=; cards; Mühlenfließ LaUna ZielonaGóra run;</td>
<td>Note: The BASECHAR function uses Unicode character decomposition mappings, so the sharp s, ß, is valid. <a href="http://www.unicode.org/reports/tr44/">http://www.unicode.org/reports/tr44/</a> #Character_Decomposition_Mappings Muhlenfließ LaUna ZielonaGóra</td>
</tr>
</tbody>
</table>
### Statements

- **Note:** The SAS session encoding is wlatin1 and the ESC option is specified.

```sas
data cities; length name$24 name_ascii$80;  
  input name;  
  name_ascii = basechar(name, "ESC"); put name_ascii=;  
cards;  
Mühlenfließ  
LaUña  
ZielonaGóra  
run;
```

- **Note:** The SAS session encoding is wlatin1 and the NCR option is specified.

```sas
data cities; length name$24 name_ascii$80;  
  input name;  
  name_ascii = basechar(name, "NCR"); put name_ascii=;  
cards;  
Mühlenfließ  
LaUña  
ZielonaGóra  
run;
```

### ENCODCOMPAT Function

Verifies the transcoding compatibility between two encodings.

**Category:** Encoding

**Syntax**

```sas
ENCODCOMPAT(source1,<source2>)
```

**Required Arguments**

- **source1**
  - a character string that represents an encoding.

- **source2**
  - a character string that represents an encoding. This argument is optional.

**Details**

If you specify one encoding, the function verifies the compatibility of the specified encoding with the current SAS session encoding.

If you specify two encodings, the function verifies the compatibility of the two encodings.

The function compares two encoding identifiers and determines whether the data needs to be transcoded. `source1` is the source encoding. `source2` is the destination encoding. Transcoding 7-bit ASCII to another type of ASCII is compatible, but transcoding ASCII to 7-bit ASCII might not be compatible.
The ENCODCOMPAT function specifies the following values:

-1  Source1 is not a valid encoding name.
-2  Source2 is not a valid encoding name.
0   The encodings are not compatible. Transcoding is needed.
1   The encodings are compatible. Transcoding is not needed.
2   A newline character is detected.

Example

The following examples demonstrate the ENCODCOMPAT features:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>/* session encoding is wlatin1 */ isCompat= EncodCompat(&quot;xyz&quot;); put isCompat;</td>
<td>-1</td>
</tr>
<tr>
<td>/* session encoding is wlatin1 */ isCompat= EncodCompat (&quot;cp1252&quot;); put isCompat;</td>
<td>1</td>
</tr>
<tr>
<td>isCompat= EncodCompat (&quot;ebcdic1149&quot;,&quot;open_ed-1149&quot;); put isCompat;</td>
<td>2</td>
</tr>
<tr>
<td>isCompat= EncodCompat (&quot;cp1251&quot;,&quot;ebcdic1149&quot;); put isCompat;</td>
<td>0</td>
</tr>
</tbody>
</table>

ENCODISVALID Function

Verifies a valid encoding name.

Category: Encoding

Syntax

ENCODISVALID(source)

Required Argument

source

a character string that represents an encoding name.

Details

The ENCODISVALID function returns the following values:

0   the character string is not a valid encoding name.
1 the character string is a valid short encoding name.
2 the character string is a valid long encoding name.
3 the character string is a valid alias encoding name.

**Example**

The following examples demonstrate the ENCODISVALID features:

<table>
<thead>
<tr>
<th>SAS Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>isValid= EncodIsValid(&quot;xyz&quot;); put isValid;</td>
<td>0</td>
</tr>
<tr>
<td>isValid= EncodIsValid(&quot;wlt2&quot;); put isValid;</td>
<td>1</td>
</tr>
<tr>
<td>isValid= EncodIsValid(&quot;wlatin2&quot;); put isValid;</td>
<td>2</td>
</tr>
<tr>
<td>isValid= EncodIsValid(&quot;cp1250&quot;); put isValid;</td>
<td>3</td>
</tr>
</tbody>
</table>

**GETLOCENV Function**

Returns the current locale/language environment.

**Category:** Locale

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

`GETLOCENV()`

**Details**

The GETLOCENV function returns the locale/language environment value for a valid SAS locale. The following environment values are possible:

**SBCS**

The SAS session encoding is SBCS (Single-Byte Character Set). SASWZSD is loaded for string manipulation.

**DBCS**

The SAS session encoding is DBCS (Double-Byte Character Set). SASWZSD is loaded for string manipulation.
MBCS

The SAS session encoding is Unicode(UTF8). SASWZSU is loaded for string manipulation.

If you receive a blank value, then the WZSS subsystem is not available. This action suggests a configuration or installation error.

Example

In the following example, the LOCALE= system option is set to French_France.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france; environ=getlocenv(); put environ;</td>
<td>SBCS</td>
</tr>
</tbody>
</table>

GETPXLANGUAGE Function

Returns the current two-letter language code.

Category: Locale

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Syntax

GETPXLANGUAGE()

Details

The GETPXLANGUAGE function returns the two-letter language code based on the current value of the LOCALE= SAS system option. The length of the language name is two characters. If the size of the variable that receives the value is less than two characters, the value is truncated.

Example

In the first example, the LOCALE= system option is set to French_France. The second example is set to German. The third example is set to English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france; lang=getpxLanguage(); put lang;</td>
<td>fr</td>
</tr>
</tbody>
</table>
See Also

System Options:

- “LOCALE System Option” on page 591

Functions:

- “GETPXREGION Function” on page 341
- “GETPXLOCALE Function” on page 340

GETPXLOCALE Function

Returns the POSIX locale value for a SAS locale.

**Category:** Locale

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

GETPXLOCALE(<source>)

**Required Argument**

<source> is an optional argument that specifies a locale name.

**Details**

The GETPXLOCALE function returns the POSIX locale value for a valid SAS locale name. If you specify an invalid locale name, then a null string is returned. If you do not specify a value for the <source> argument, then the function returns the POSIX name for the current SAS session. The length of the POSIX locale name is five characters. If the size of the variable that receives the value is less than five characters, the value is truncated.
Example

In the first example, the LOCALE= system option is set to French_France. In the second example, the \texttt{<source>} argument is set to German_Germany. In the third example, the \texttt{<source>} argument is set to English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france;</td>
<td>fr_FR</td>
</tr>
<tr>
<td>locale=getpxLocale();</td>
<td></td>
</tr>
<tr>
<td>put locale;</td>
<td></td>
</tr>
<tr>
<td>getpxLocale(&quot;german_germany&quot;);</td>
<td>de_DE</td>
</tr>
<tr>
<td>put locale;</td>
<td></td>
</tr>
<tr>
<td>getpxLocale(&quot;english_unitedstates&quot;);</td>
<td>en_US</td>
</tr>
<tr>
<td>put locale;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

System Options:

- “LOCALE System Option” on page 591

Functions:

- “GETPXLANGUAGE Function” on page 339
- “GETPXREGION Function” on page 341

GETPXREGION Function

Returns the current two-letter region code.

\textbf{Category:} Locale

\textbf{Restriction:} This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

\textbf{Syntax}

\texttt{GETPXREGION()}

\textbf{Details}

The GETPXREGION function returns the two-letter region code based on the current LOCALE= SAS system option. The length of the region name is two characters. If the size of the variable that receives the value is less than two characters, the value is truncated.
Example

In the first example the LOCALE= system option is set to French_France. The second example is set to German. The third example is set to English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france;</td>
<td>FR</td>
</tr>
<tr>
<td>region=getpxRegion();</td>
<td></td>
</tr>
<tr>
<td>put region;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>option locale=German;</td>
<td>DE</td>
</tr>
<tr>
<td>region=getpxRegion();</td>
<td></td>
</tr>
<tr>
<td>put region;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>option locale=en_US;</td>
<td>US</td>
</tr>
<tr>
<td>region=getpxRegion();</td>
<td></td>
</tr>
<tr>
<td>put region;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

System Options:

- “LOCALE System Option” on page 591

Functions:

- “GETPXLOCALE Function” on page 340
- “GETPXLANGUAGE Function” on page 339

KCOMPARE Function

Returns the result of a comparison of character expressions.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Tip:** Non-DBCS equivalent function is the “COMPARE Function” in SAS Functions and CALL Routines: Reference.

**Syntax**

\[ \text{KCOMPARE(source, pos, count, findstr)} \]
Required Arguments

*source*
   specifies the character expression to be compared.

*pos*
   specifies the starting position in *source* to begin the comparison. If *pos* is omitted, the entire *source* is compared. If *pos* is less than 0, *source* is assumed as extended DBCS data that does not contain any SO/SI characters.

*count*
   specifies the number of bytes to compare. If *count* is omitted, all of *source* that follows *pos* is compared, except for any trailing blanks.

*findstr*
   specifies the character expression to compare to *source*.

Details

KCOMPARE returns values as follows:

- a negative value if *source* is less than *findstr*
- 0 if *source* is equal to *findstr*
- a positive value if *source* is greater than *findstr*

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options pageno=1 nodate ls=80 ps=80;</td>
<td>The SAS System</td>
</tr>
<tr>
<td>data test;</td>
<td></td>
</tr>
<tr>
<td>rc1 = kcompare('漢字のテスト','漢字');</td>
<td>Obs rc1 rc2 rc3 rc4 rc5</td>
</tr>
<tr>
<td>rc2 = kcompare('漢字のテスト',7,'テスト');</td>
<td>1 3 0 3 1 4</td>
</tr>
<tr>
<td>rc3 = kcompare('漢字のテスト',1,8,'漢字');</td>
<td></td>
</tr>
<tr>
<td>rc4 = kcompare('漢字のテスト','ABC');</td>
<td></td>
</tr>
<tr>
<td>rc5 = kcompare('ABCDEF','ABC');</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>proc print data=test; run;</td>
<td></td>
</tr>
</tbody>
</table>

KCOMPRESS Function

Removes specified characters from a character expression.

Category: DBCS
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Tip: Non-DBCS equivalent function is COMPARE in SAS Functions and CALL Routines: Reference.

Syntax

KCOMPRESS(source,<characters-to-remove>)

Required Arguments

source
specifies a character expression that contains the characters to be removed. When only source is specified, KCOMPRESS returns this expression with all of the single and double-byte blanks removed.

characters-to-remove
specifies the character or characters that KCOMPRESS removes from the character. If characters-to-remove is omitted, KCOMPRESS removes all blanks.

Tip Enclose a literal string of characters in quotation marks.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>result=kcompress('漢字のテスト',')');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>result=漢字テスト</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:

- “KLEFT Function” on page 351
- “KTRIM Function” on page 370

KCOUNT Function

Returns the number of double-byte characters in an expression.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.
Syntax

\texttt{KCOUNT(source)}

\textbf{Required Argument}

\textit{source}

specifies the character expression to count.

\textbf{Details}

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

\textbf{Example}

The following example uses Japanese characters.

\begin{tabular}{|l|l|}
\hline
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{data _null;} & \texttt{result=3} \\
\texttt{\hspace{1em}text='abcあいう';} & \\
\texttt{\hspace{1em}result=kcount(text);} & \\
\texttt{\hspace{1em}put result=;} & \\
\texttt{\hspace{1em}run;} & \\
\hline
\end{tabular}

\textbf{KCVT Function}

Converts data from one type of encoding data to another type of encoding data.

\textbf{Category:} Character

\textbf{Restriction:} This function is assigned an I18N Level 1 status. If possible, avoid I18N Level 1 functions if you are using a non-English language. Under certain circumstances, the I18N Level 1 functions might not work correctly with Double-Byte Character Set (DBCS) or Multi-Byte Character Set (MBCS) encodings. For more information, see Internationalization Compatibility on page 307.

\textbf{Syntax}

\texttt{KCVT(text, intype, outype, <options,...>)}

\textbf{Required Arguments}

\textit{text}

specifies the character variable to be converted.
intype
specifies the encoding of the data. The encoding of the text must match the input data's encoding. For valid values, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

ASCIIANY and EBCIDICANY are invalid encoding values.

outtype
specifies the encoding to be converted into character data. For valid values, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

ASCIIANY and EBCIDICANY are invalid encoding values.

options
specifies character data options. Here are the available options:

- **NOSOSI** | **NOSHIFT**
  No shift code or Hankaku characters.

- **INPLACE**
  Replaces character data by conversion. The INPLACE option is specified to secure the same location between different hosts whose lengths of character data are not identical. For example, the INPLACE option converts data from the host that requires Shift-Codes into the other host, which does not require Shift Codes. Truncation occurs when the length of the character data that is converted into outtype for Shift-Codes is longer than the length that is specified in intype.

- **KANA**
  Includes Hankaku katakana characters in columns of character data.

- **UPCASE**
  Converts a 2-byte alphabet to uppercase characters.

- **LOWCASE**
  Converts a 2-byte alphabet to lowercase characters.

- **KATA2HIRA**
  Converts katakana data to hiragana.

- **HIRA2KATA**
  Converts hiragana data to katakana.

Details
See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KCVT function converts SBCS, DBCS, and MBCS character strings into encoding data. For example, the KCVT function can convert the following: ASCII code data to UCS2 encoding data, Greek code data to UTF-8, and Japanese SJIS code data to another Japanese code data. You can specify the following types for the Intype and Outtype options: UCS2, UCS2L, UCS2B, and UTF8. To enable the DBCS mode, specify the DBCS option in the configuration file or from the command line.

If the KCVT function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

Example
The following code converts IBM PC codes into DEC codes for the external text file specified as *my-input-file* and writes the results in OUTDD.

```sas
data _null_;
  infile ‘my-input-file’;
```

file outdd noprint;
input @1 text $char80.;
text = kcvt(text, 'pcibm', 'dec');
put @1 text $char80.;
run;

See Also

System options:
• “DBCS System Option: UNIX, Windows, and z/OS” on page 581

Procedure:
• Chapter 17, “DBCSTAB Procedure,” on page 649

KINDEX Function

Searches a character expression for a string of characters.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Tip: Non-DBCS equivalent function is INDEX in SAS Functions and CALL Routines: Reference

Syntax

KINDEX(source, excerpt)

Required Arguments

source
specifies the character expression to search.

excerpt
specifies the string of characters to search for in the character expression.

Tip Enclose a literal string of characters in quotation marks.

Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KINDEX function searches source, from left to right, for the first occurrence of the string that is specified in excerpt, and returns the position in source of the string’s first character. If the string is not found in source, KINDEX returns a value of 0. If there are multiple occurrences of the string, KINDEX returns only the position of the first occurrence.
The following example uses Japanese characters.

```plaintext
data _null;
text='漢字文字列の検索';
result=kindex(text,'検索');
put result=;
text2='探す';
result=kindex(text,text2);
put result=;
run;
```

See Also

Functions:
- “KINDEXC Function” on page 350

KINDEXB Function

Searches a character expression for a string of characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

KINDEXB(source, excerpt)

**Required Arguments**

- **source**
  - specifies the character expression to search for.

- **excerpt**
  - specifies the string of characters to search for in the character expression.

  **TIP** Enclose a literal string of characters in quotation marks.

**Details**

The KINDEXB function searches `source`, from left to right, for the first occurrence of the string that is specified in `excerpt` and returns the string's first character position in `source`. If the string is not found in `source`, KINDEXB returns a value of 0. If there are multiple occurrences of the string, KINDEXB returns only the position of the first occurrence.
Comparisons

KINDEXB returns byte-based values. KINDEX returns character-based values. When processing an SBCS string (for example, wlatin1 encoding), KINDEXB and KINDEX return identical results. But in a DBCS session (for example, EUC-JP or SHIFT-JIS encoding), most CJK characters are 2 bytes wide, causing these functions to return different results.

The KINDEXCB function searches for the first occurrence of any individual character that is present within the character string. The KINDEXB function searches for the first occurrence of the character string as a pattern.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>text='漢字文字列の検索';</td>
<td>Result=13</td>
</tr>
<tr>
<td>result=kindexb(text,'検索');</td>
<td>Result=0</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>text2='探';</td>
<td></td>
</tr>
<tr>
<td>result=kindexb(text2);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KINDEXCB Function

Searches a character expression for specified characters.

- **Category:** DBCS
- **Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

\[
\text{KINDEXCB}(\text{source, excerpt-1, ..., excerpt-n})
\]

**Required Arguments**

- **source**
  - specifies the character expression to search for.

- **excerpt-1, ..., excerpt-n**
  - specifies the characters to search for in the character expression.

**Tips**

If you specify more than one excerpt, separate them with a comma. Enclose a literal string of characters in quotation marks.
Details

The KINDEXCB function searches the data in source, from left to right, for the first occurrence of any character in the excerpts’ arguments and returns the position in source of that character. If none of the characters in excerpt-1 through excerpt-n in source are found, KINDEXCB returns a value of 0.

Comparisons

KINDEXCB returns byte-based values. KINDEXC returns character-based values. When processing SBCS strings (for example, wla1 encoding), they return identical results. But in a DBCS session (for example, EUC-JP or SHIFT-JIS encoding), most CJK characters are 2 bytes wide, causing these functions to return different results.

The KINDEXCB function searches for the first occurrence of any individual character that is present within the character string, whereas the KINDEXB function searches for the first occurrence of the character string as a pattern.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>Result=13</td>
</tr>
<tr>
<td>a='A B C. def [漢字123456]';</td>
<td></td>
</tr>
<tr>
<td>result = kindexcb(a, '1234567890', '漢漢');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KINDEXC Function

Searches a character expression for specified characters.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Tip: The Non-DBCS equivalent function is “INDEXC Function” in SAS Functions and CALL Routines: Reference

Syntax

KINDEXC(source, excerpt-1, ...excerpt-n)

Required Arguments

source
specifies the character expression to search for.

excerpt-1, ...excerpt-n
specifies the characters to search for in the character expression.
Tips  If you specify more than one excerpt, separate them with a comma.

Enclose a literal string of characters in quotation marks.

Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KINDEXC function searches source, from left to right, for the first occurrence of any character present in the excerpts and returns the position in source of that character. If none of the characters in excerpt-1 through excerpt-n in source are found, KINDEXC returns a value of 0.

Comparisons

The KINDEXC function searches for the first occurrence of any individual character that is present within the character string, whereas the KINDEX function searches for the first occurrence of the character string as a pattern.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=9</td>
</tr>
<tr>
<td>a=’ＡＢＣ．def（漢字123456）’;</td>
<td></td>
</tr>
<tr>
<td>result=kindexc(a,’1234567890’,’感漢’);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Function:

• “KINDEX Function” on page 347

KLEFT Function

Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/Sl.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Tip: Non-DBCS equivalent function is LEFT in SAS Functions and CALL Routines: Reference.
Syntax

KLEFT(argument)

Required Argument

argument

specifies any SAS character expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

KLEFT returns an argument and removes the leading blanks.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; add proposals;</td>
<td></td>
</tr>
<tr>
<td>a=’漢字 の テスト’;</td>
<td>漢字 の テスト</td>
</tr>
<tr>
<td>b=kleft(a);</td>
<td></td>
</tr>
<tr>
<td>put ’----------1------------2-----’;</td>
<td>漢字 的 テスト</td>
</tr>
<tr>
<td>put a $25.;</td>
<td></td>
</tr>
<tr>
<td>put b $25.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:

- “KCOMPRESS Function” on page 343
- “KRIGHT Function” on page 361
- “KTRIM Function” on page 370

KLENGTH Function

Returns the length of an argument.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Tip: Non-DBCS equivalent function is LENGTH in SAS Functions and CALL Routines: Reference.
**Syntax**

\[ \text{KLENGTH}(\text{argument}) \]

**Required Argument**

\[ \text{argument} \]

specifies any SAS expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KLENGTH function returns an integer that represents the position of the rightmost non-blank character in the argument. If the value of the argument is missing, KLENGTH returns a value of 0. If the argument is an uninitialized numeric variable, KLENGTH returns a value of 12 and prints a note in the SAS log that the numeric values have been converted to character values.

**Example**

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=8</td>
</tr>
<tr>
<td>text='abcあいう';</td>
<td></td>
</tr>
<tr>
<td>result=klength(text);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

---

**KLOWCASE Function**

Converts all letters in an argument to lowercase.

- **Category:** DBCS
- **Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.
- **Tip:** Non-DBCS equivalent function is LOWCASE in SAS Functions and CALL Routines: Reference.

**Syntax**

\[ \text{KLOWCASE}(\text{argument}) \]
**Required Argument**

*argument*

specifies any SAS character expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KLOWCASE function copies a character argument, converts all uppercase letters to lowercase letters, and returns the altered value as a result.

**Example**

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=abcあいうえお</td>
</tr>
<tr>
<td>result=klowcase('ABCあいうえお');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**KPROPCASE Function**

Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see *Internationalization Compatibility on page 307.*

**Syntax**

\[ str=\text{KPROPCASE}(<\text{instr}> , (<\text{options}>) ) \]

**Required Arguments**

*str*

data string that has been converted and is in the current SAS session encoding.

*instr*

input data string.

*options*

converts Japanese, Chinese, Korean, and Taiwanese characters based on specified options.

**HALF-KATAKANA, FULL-KATAKANA**

This option converts half-width katakana to full-width katakana and is used only with Japanese encoding.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FULL-KATAKANA, HALF-KATAKANA</strong></td>
<td>This option converts full-width katakana to half-width katakana and is used only with Japanese encoding.</td>
<td>This option cannot be used at the same time with the half-Katakana, full-Katakana option.</td>
</tr>
<tr>
<td><strong>KATAKANA, ROMAJI</strong></td>
<td>This option converts the katakana character string to a romaji character string and is used only with Japanese encoding.</td>
<td>This option cannot be used at the same time with the romaji, katakana option.</td>
</tr>
<tr>
<td><strong>ROMAJI, KATAKANA</strong></td>
<td>This option converts the romaji character string to a katakana character string and is used only with Japanese encoding.</td>
<td>This option cannot be used at the same time with the katakana, romaji option.</td>
</tr>
<tr>
<td><strong>FULL-ALPHABET, HALF-ALPHABET</strong></td>
<td>This option converts the Full-Alphabet characters to Half-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.</td>
<td>This option cannot be used at the same time with the Half-Alphabet, Full-Alphabet option.</td>
</tr>
<tr>
<td><strong>HALF-ALPHABET, FULL-ALPHABET</strong></td>
<td>This option converts the Half-Alphabet characters to Full-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.</td>
<td>This option cannot be used at the same time with the Full-Alphabet, Half-Alphabet option.</td>
</tr>
<tr>
<td><strong>LOWERCASE, UPPERCASE</strong></td>
<td>This option converts lowercase alphabet characters to uppercase alphabet characters.</td>
<td>This option cannot be used at the same time with the Uppercase, Lowercase option.</td>
</tr>
<tr>
<td><strong>UPPERCASE, LOWERCASE</strong></td>
<td>This option converts uppercase alphabet characters to lowercase alphabet characters.</td>
<td>This option cannot be used at the same time with the Lowercase, Uppercase option.</td>
</tr>
<tr>
<td><strong>PROPER</strong></td>
<td>This option specifies the following default options based on the encoding:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Japanese encoding</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Half-Katakana,Full-Katakana</td>
<td></td>
</tr>
</tbody>
</table>
• Full-alphabet, Half-alphabet
• Lowercase, Uppercase
• Korean encoding:
  • Full-alphabet, Half-alphabet
• Chinese encoding:
  • Full-alphabet, Half-alphabet
• Taiwanese encoding:
  • Full-alphabet, Half-alphabet

Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

This function converts the input string based on the specified options and default options. The KPROPCASE function supports the Chinese, Japanese, Korean, Taiwanese (CJKT) environment.

Example

The following example demonstrates the functionality of the KPROPCASE function:

```
length fullkana halfkana upper lower fullalpha $ 200;
length str1 str2 str3 str4 str5 str7 str8 $ 30 str6 $44;
lower = 'do-naxtutsu'; /* Doughnuts in Japanese Roman word. */
upper = 'DO-NAXTUTSU'; /* Doughnuts in Japanese Roman word. */
fullkana = unicode('\u30C9\u30FC\u30CA\u30C3\u30C4');
halfkana = unicode('\uFF84\uFF9E\uFF70\uFF85\uFF6F\uFF82');
fullalpha = unicode('\uFF24\uFF2F\uFF0D\uFF2E\uFF21\uFF38\uFF34\uFF35\uFF33\uFF35');
str1 = kpropcase(fullkana, 'full-katakana,half-katakana');
if (halfkana EQ trim(str1)) then
  put str1= $hex14.;
str2 = kpropcase(halfkana, 'half-katakana, full-katakana');
if (fullkana EQ trim(str2)) then
  put str2= $hex22.;
str3 = kpropcase(fullkana, 'katakana,romaji');
if (trim(str3) EQ upper) then
  put str3= ;
str4 = kpropcase(upper, 'romaji,katakana');
if (trim(str4) EQ fullkana) then
  put str4= $hex22.;
str5 = kpropcase(fullalpha, 'full-alphabet, half-alphabet');
if (trim(upper) EQ str5) then
  put str5=;
str6 = kpropcase(upper, 'half-alphabet, full-alphabet');
if (trim(str6) EQ fullalpha) then
  put str6= $hex46.;
str7 = kpropcase(lower, 'lowercase, uppercase');
if (trim(str7) EQ upper) then
  put str7=;
str8 = kpropcase(upper, 'uppercase, lowercase');
if (trim(str8) EQ lower) then
```

put str8=;
RESULTS:
str1=C4DEB0C5AFC220
str2=8368815B83698362836320
str3=DO-NAXTUTSU
str4=8368815B83698362836320
str5=DO-NAXTUTSU
str6=8263826E817C826D826082778273827482738272827420
str7=DO-NAXTUTSU
str8=do-naxtutsu

KPROPCHAR Function

Converts special characters to normal characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

\[ str = \text{KPROPCHAR}(<\text{instr}>) \]

**Required Arguments**

\textit{str}

result string. Special characters are converted to normal characters.

\textit{instr}

input data string.

**Details**

This function converts special characters to normal characters. The KPROPCHAR function converts the characters from the following ranges:


**Example**

The following example demonstrates the functionality of the KPROPCHAR function. These examples are processed in a UTF8 SAS session:

```sas
length in1 out1 $30 ;
   in1=unicode("\u2460\u2473\u277F\u325F");
   out1=KPROPCHAR(in1);
   put out1;
```
RESULTS:
(1)(20)(-10)(35)

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>; length in1 out1 $30 ; in1=unicode('\u2460\u2473\u277F'); out1=KPROPCHAR(in1); put out1; run;</td>
<td>(1)(20)(-10)(35)</td>
</tr>
<tr>
<td>data <em>null</em>; length in1 out1 $30 ; in1=unicode('\u2776\u2793'); out1=KPROPCHAR(in1); put out1; run;</td>
<td>(-1)(-10)</td>
</tr>
<tr>
<td>data <em>null</em>; length in1 out1 $30 ; in1=unicode('\u3200\u32FE'); out1=KPROPCHAR(in1); put out1; run;</td>
<td>(☐)(☒)</td>
</tr>
</tbody>
</table>

**KPROPDATA Function**

Removes or converts unprintable characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see *Internationalization Compatibility* on page 307.

**Syntax**

\[
\text{str} = \text{KPROPDATA}(\text{<instr>}, \text{<options>}, \text{<input encode name>}, \text{<output encode name>})
\]

**Required Arguments**

- **str**
  - data string that has been converted and is in session encoding.

- **instr**
  - input data string.

- **options**
  - specifies instructions on processing unprintable characters:
    - **UESC**
      - Converts unprintable characters using a Unicode escaped string (for example, \u0000\u1234).
TRIM
   Removes unprintable characters. No replacement character is used.

BLANK or ' '  
   Replaces each unprintable character with a single-byte blank.

QUESTION or '?'  
   Replaces unprintable characters with a single-byte '?'.

HEX  
   Replaces unprintable characters with a hexadecimal representation (for example, 0x810x82).

TRUNCATE or TRUNC  
   Truncates the data string when the first unprintable character is encountered.

REMOVE  
   Removes the data string if any unprintable characters are found.

NCR  
   Encodes the unprintable characters using NCR representation if the code is available in Unicode.

input encode name  
   specifies the input data's encoding name if necessary. If the input encode name is not specified, then the KPROPDATA function processes the data as the current SAS session encoded string. For information about SAS encoding names, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

output encode name  
   specifies the output data's encoding name. If the encoding name is not specified, the KPROPDATA function recognizes the output as the current SAS session encoding. For information about SAS encoding names, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

Details

This function converts the input data string to the current SAS session encoding and removes or replaces unprintable characters based on the options.

Example

The following example demonstrates the functionality of the KPROPDATA function:

```sas
length instr $12;
length str1 str2 str3 str4 str5 str6 str7 str8 str9 str10$ 50;
instr = "534153"x || "ae"x || " System";
put instr;
str1 = kpropdata(instr);
put str1= +2 str1= $hex26.;
str2 = kpropdata(instr,'UESC');
put str2= +2 str2= $hex26.;
str3 = kpropdata(instr, 'UESC','wlatin1');
put str3= +2 str3= $hex34.;
str4 = kpropdata(instr,'TRIM','wlatin1');
put str4= +2 str4= $hex26.;
str5 = kpropdata(instr,'BLANK', 'wlatin1');
put str5= +2 str5= $hex26.;
str6 = kpropdata(instr,'?', 'wlatin1');
put str6= +2 str6= $hex26.;
```
str7 = kpropdata(instr,'hex', 'wlatin1');
put str7= +2 str7= $hex26.;;
str8 = kpropdata(instr,'TRUNC', 'wlatin1');
put str8= +2 str8= $hex26.;;
str9 = kpropdata(instr,'REMOVE', 'wlatin1');
put str9= +2 str9= $hex26.;;
str10 = kpropdata(instr,'NCR', 'wlatin1');
put str10= +2 str10= $hex26.;;

RESULTS:
SAS? System
str1=SAS? System  str1=534153AE2053797374656D2020
str2=SAS? System  str2=534153AE2053797374656D2020
str3=SAS\uff6e System  str3=5341535C75666536652053797374656D20
str4=SAS System  str4=5341532053797374656D202020
str5=SAS System  str5=534153202053797374656D2020
str6=SAS System  str6=5341533F2053797374656D2020
str7=SAS\xAE System  str7=5341535C784145205379737465
str8=SAS  str8=534153202020202020202020202020
str9=  str9=2020202020202020202020202020202020
str10=SAS® System  str10=53415326233137343B20537973

**KREVERSE Function**

Reverses a character expression.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see [Internationalization Compatibility on page 307](#).

**Tip:** Non-DBCS equivalent function is `REVERSE` in [SAS Functions and CALL Routines: Reference](#).

**Syntax**

\[ \text{KREVERSE}(\text{argument}) \]

**Required Argument**

`argument` specifies any SAS character expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

**Example**

The following example uses Japanese characters.
KRIGHT Function

Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Tip:** See “RIGHT Function” in SAS Functions and CALL Routines: Reference.

**Syntax**

\[
\text{KRIGHT}(\text{argument})
\]

**Required Argument**

\[\text{argument}\]

specifies any SAS character expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KRIGHT function returns an argument with trailing blanks moved to the start of the value. The argument's length does not change.

**Example**

The following example uses Japanese characters.

```sas
data _null_;  
  result=kreverse('漢字のテスト');  
  put result=;  
run;
```
**KSCAN Function**

Selects a specified word from a character expression.

**Category:** DBCS  
**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Tip:** Non-DBCS equivalent function is SCAN in SAS Functions and CALL Routines: Reference.

**Syntax**

\[
\text{KSCAN}(\text{argument}, n, \text{delimiters})
\]

**Required Arguments**

- **argument**
  - specifies any character expression.

- **n**
  - specifies a numeric expression that produces the number of the word in the character expression that you want KSCAN to select.

  **Tip:** If \(n\) is negative, KSCAN selects the word in the character expression starting from the end of the string. If \(|n|\) is greater than the number of words in the character expression, KSCAN returns a blank value.

- **delimiters**
  - specifies a character variable that produces characters that you want KSCAN to use as word separators in the character expression.

---

```sas
data _null_;  
a='私のテスト';  
b=right(a);  
put '-----1-----2-----';  
put a $25.;  
put b $25.;  
run;
```

---

**See Also**

Functions:
- “KCOMPRESS Function” on page 343
- “KLEFT Function” on page 351
- “KTRIM Function” on page 370
Defaults
If you omit delimiters in an ASCII environment, SAS uses blank . < ( + & ! $ * ) : ^ – / , % | . In ASCII environments without the ^ character, KSCAN uses the – character instead.

If you omit delimiters on an EBCDIC environment, SAS uses blank . < ( + | & ! $ * ) ; ¬ – / , % | ¢

Tip
If you represent delimiters as a constant, enclose delimiters in quotation marks.

Details
See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

Leading delimiters before the first word in the character string do not effect KSCAN. If there are two or more contiguous delimiters, KSCAN treats them as one.

Example
The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>length x $20 y $20;</td>
<td></td>
</tr>
<tr>
<td>text1='これは漢字関数のテストです。';</td>
<td></td>
</tr>
<tr>
<td>x='norblank';</td>
<td></td>
</tr>
<tr>
<td>i=1;</td>
<td></td>
</tr>
<tr>
<td>do until(x=' ');</td>
<td>x=これ</td>
</tr>
<tr>
<td>x=kscan(text1,i,'はの');</td>
<td>x=漢字関数</td>
</tr>
<tr>
<td>put x=;</td>
<td>x=テストです。</td>
</tr>
<tr>
<td>i=i+1;</td>
<td></td>
</tr>
<tr>
<td>end;</td>
<td></td>
</tr>
<tr>
<td>y='norblank';</td>
<td></td>
</tr>
<tr>
<td>i=-1;</td>
<td>y=テストです。</td>
</tr>
<tr>
<td>do until(y=' ');</td>
<td></td>
</tr>
<tr>
<td>y=kscan(text1,i,'はの');</td>
<td>y=漢字関数</td>
</tr>
<tr>
<td>put y=;</td>
<td>y=これ</td>
</tr>
<tr>
<td>i=i-1;</td>
<td></td>
</tr>
<tr>
<td>end;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

KSTRCAT Function
Concatenates two or more character expressions.

Category: DBCS
KSTRCAT Function

Removes leading and trailing blanks from a character string.

**Category:** Character

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

### Syntax

```
KSTRCAT(argument-1, argument-2<, … argument-n>)
```

### Required Argument

*argument*

specifies any single-byte or double-byte character expression.

### Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

KSTRCAT concatenates two or more single-byte or double-byte character expressions. It also removes unnecessary SO/SI pairs between the expressions.

If the KSTRCAT function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

### Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>text1='漢字';</td>
<td>result='漢字文字列の連結'</td>
</tr>
<tr>
<td>text2='文字列';</td>
<td></td>
</tr>
<tr>
<td>text3='連結';</td>
<td></td>
</tr>
<tr>
<td>result=kstrcat(text1,text2,'の',text3);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
Syntax

KSTRIP(string)

Required Argument

string

specifies a character constant, variable, or expression.

Details

The KSTRIP function returns the argument with all leading and trailing SBCS blanks removed. If the argument is blank, KSTRIP returns a string with a length of zero.

Assigning the results of KSTRIP to a variable does not affect the length of the receiving variable. If the value that is trimmed is shorter than the length of the receiving variable, SAS pads the value with new trailing blanks.

In a DATA step, if the KSTRIP function returns a value to a variable that has not previously been assigned a length, then that variable is given the length of the argument.

Comparisons

The KSTRIP and STRIP functions return a string with a length of zero, for strings that are blank, and the TRIM function returns a single blank.

The KSTRIP and STRIP functions truncate the output if the receiving variable length is less than the input string. The STRIP function processes only SBCS string truncation. The KSTRIP function processes SBCS and DBCS data.

KSTRIP and STRIP functions are used for data normalization purposes. The KSTRIP function should be used in a DBCS environment. The KSTRIP and STRIP functions have similar performance actions.

Example

The following example shows how the KSTRIP function deletes leading and trailing blanks, and how the DBCS character is truncated. This example uses the Japanese Shift_JIS encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>93B893B820</td>
</tr>
<tr>
<td>length a $5;</td>
<td></td>
</tr>
<tr>
<td>v = unicode(’\u0020\u0020\u8404\u8404\u8404\u0020\u0020’);</td>
<td></td>
</tr>
<tr>
<td>put v = hex.;</td>
<td></td>
</tr>
<tr>
<td>a = kstrip(v);</td>
<td></td>
</tr>
<tr>
<td>put a = hex.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Function

- “STRIP Function” in SAS Functions and CALL Routines: Reference

KSUBSTR Function

Extracts a substring from an argument.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Tip:** See “SUBSTR (left of =) Function” in SAS Functions and CALL Routines: Reference.

**Syntax**

KSUBSTR(argument, position<n>)

**Required Arguments**

- **argument**  
  specifies any SAS character expression.

- **position**  
  specifies a numeric expression that is the beginning character position.

- **n**  
  specifies a numeric expression that is the length of the substring to extract.

**Interaction**  
If $n$ is larger than the length of the expression that remains in argument after position, SAS extracts the remainder of the expression.

**Tip**  
If you omit $n$, SAS extracts the remainder of the expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KSUBSTR function returns a portion of an expression that you specify in argument. The portion begins with the character specified by position and is the number of characters specified by $n$.

A variable that is created by KSUBSTR obtains its length from the length of argument.

**Example**

The following example uses Japanese characters.
### KSUBSTRB Function

Extracts a substring from an argument according to the byte position of the substring in the argument.

**Category:** DBCS  
**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

#### Syntax

\[
\text{KSUBSTRB}(\text{argument}, \text{position}<,n>)
\]

#### Required Arguments

- **argument**  
  specifies any SAS character expression.

- **position**  
  specifies the beginning character position in byte units.

- **n**  
  specifies the length of the substring to extract in byte units.

**Interaction**

If \(n\) is larger than the length (in byte units) of the expression that remains in `argument` after `position`, SAS extracts the remainder of the expression.

**Tip**  
If you omit \(n\), SAS extracts the remainder of the expression.

### See Also

**Function:**
- “KSUBSTRB Function” on page 367

```sas
data _null_
  text='漢字文字列の抽出';  
result=ksubstr(text,1,2);  
put result=;  
result=ksubstr(text,3,4);  
put result=;  
kstart=7;  
klen=2;  
result=ksubstr(text,kstart,klen);  
put result=;  
run;
```
Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KSUBSTRB function returns a portion of an expression that you specify in argument. The portion begins with the byte unit specified by position and is the number of byte units specified by \( n \).

A variable that is created by KSUBSTRB obtains its length from the length of argument.

Example

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>text='漢字文字列の抽出';</td>
<td>result=漢字文</td>
</tr>
<tr>
<td>result=ksubstrb(text,1,6);</td>
<td>result=字文</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>result=ksubstrb(text,3,5);</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Function:
- “KSUBSTR Function” on page 366

KTRANSLATE Function

Replaces specific characters in a character expression.

<table>
<thead>
<tr>
<th>Category:</th>
<th>DBCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.</td>
</tr>
<tr>
<td>z/OS specifics:</td>
<td>This function runs only on z/OS. Uses the EBCDIC code sequence.</td>
</tr>
<tr>
<td>Tip:</td>
<td>Non-DBCS equivalent function is TRANSLATE in SAS Functions and CALL Routines: Reference.</td>
</tr>
</tbody>
</table>

Syntax

\[
\text{KTRANSLATE}(\text{source}, \text{to-1,from-1}, \text{<,…,to-n,from-n>})
\]
**Required Arguments**

*source*
- specifies the SAS expression that contains the original character value.

*to*
- specifies the characters that you want KTRANSLATE to use as substitutes.

*from*
- specifies the characters that you want KTRANSLATE to replace.

**Interaction** Values of *to* and *from* correspond on a character-by-character basis; KTRANSLATE changes character one of *from* to character one of *to*, and so on. If *to* has fewer characters than *from*, KTRANSLATE changes the extra *from* characters to blanks. If *to* has more characters than *from*, KTRANSLATE ignores the extra *to* characters.

**Note** You must have pairs of *to* and *from* arguments on some operating environments. On other operating environments, a segment of the collating sequence replaces null *from* arguments.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

You can use KTRANSLATE to translate a single-byte character expression to a double-byte character expression, or translate a double-byte character expression to a single-byte character expression.

The maximum number of pairs of *to* and *from* arguments that KTRANSLATE accepts depends on which operating environment you use to run SAS. There is no functional difference between using several pairs of short arguments, or fewer pairs of longer arguments.

If the KTRANSLATE function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

**Example**

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>result=正答</td>
</tr>
<tr>
<td>result=ktranslate('正課','回答','二課');</td>
<td>result=アイウ</td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>result=ktranslate('abc','アイウ','abc');</td>
<td></td>
</tr>
<tr>
<td>put result=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
KTRIM Function

Removes trailing DBCS blanks and SO/SI from character expressions.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Tip:** Non-DBCS equivalent function is “TRIM Function” in SAS Functions and CALL Routines: Reference.

---

**Syntax**

`KTRIM(argument)`

**Required Argument**

*argument*

specifies any SAS character expression.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

KTRIM copies a character argument, removes all trailing blanks, and returns the trimmed argument as a result. If the argument is blank, KTRIM returns one blank. KTRIM is useful for concatenating because concatenation does not remove trailing blanks.

Assigning the results of KTRIM to a variable does not affect the length of the receiving variable. If the trimmed value is shorter than the length of the receiving variable, SAS pads the value with new blanks as it assigns it to the variable.

**Example**

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>漢字のテスト</td>
</tr>
<tr>
<td>part1=’漢字の’;</td>
<td>漢字のテスト</td>
</tr>
<tr>
<td>part2=’テスト’;</td>
<td></td>
</tr>
<tr>
<td>hasblank=part1</td>
<td></td>
</tr>
<tr>
<td>noblank=ktrim(part1)</td>
<td></td>
</tr>
<tr>
<td>put hasblank;</td>
<td></td>
</tr>
<tr>
<td>put noblank;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Functions:

- “KCOMPRESS Function” on page 343
- “KLEFT Function” on page 351
- “KRIGHT Function” on page 361

KTRUNCATE Function

Truncates a string to a specified length in byte unit without breaking multibyte characters.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

\[
\text{KTRUNCATE}(\text{argument, number, length})
\]

**Required Arguments**

- **argument** specifies any SAS character expression.
- **number** is numeric.
- **length** is an integer.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KTRUNCATE function truncates a full-length number (stored as a double) to a smaller number of bytes, as specified in length and pads the truncated bytes with 0s. The truncation and subsequent expansion duplicate the effect of storing numbers in less than full length and then reading them.

**Example**

The following example uses Japanese characters.
KUPCASE Function

Converts all letters in an argument to uppercase.

Category: DBCS

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.


Syntax

KUPCASE(argument)

Required Argument

argument

specifies any SAS character expression.

Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KUPCASE function copies a character argument, converts all lowercase letters to uppercase letters, and returns the altered value as a result.

Example

The following example uses Japanese characters.

data _null_;
   x1=ktruncate('漢字のテスト',4);
   x2=ktruncate('漢字のテスト',5);
   x3=ktruncate('漢字のテスト',6);
   y1=ktruncate('漢字のテスト',3,6);
   y2=ktruncate('漢字のテスト',4,6);
   y3=ktruncate('漢字のテスト',5,6);
   put x1= / x2= / x3= / y1= / y2= / y3=;
run;
KUPDATE Function

Inserts, deletes, and replaces character value contents.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

`KUPDATE(argument, position, n<, characters-to-replace>)`

`KUPDATE(argument, position<, n>, characters-to-replace)`

**Required Arguments**

- **argument**
  - specifies a character variable.

- **position**
  - specifies a numeric expression that is the beginning character position.

- **n**
  - specifies a numeric expression that is the length of the substring to be replaced.

**Restrictions**

- **n** cannot be larger than the length of the expression that remains in `argument` after `position`.

- **n** is optional, but you cannot omit both **n** and `characters-to-replace` from the function.

**Tip**

- If you omit `n`, SAS uses all of the characters in `characters-to-replace` to replace the values of `argument`.

- `characters-to-replace` specifies a character expression that replaces the contents of `argument`.

**Restriction**

- `characters-to-replace` is optional, but you cannot omit both `characters-to-replace` and `n` from the function.

**Tip**

- Enclose a literal string of characters in quotation marks.

```plaintext
data _null_;  
  result=kupdate('abcあいうえお');  
  put result=;  
run;
```

result=ABCあいうえお
Details

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KUPDATE function replaces the value of argument with the expression in characters-to-replace starting at the character that you specify in position.

If the KUPDATE function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

Examples

Example 1

The following example shows the difference between KUPDATE and KUPDATES.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>str='123456';</td>
<td></td>
</tr>
<tr>
<td>x1=str; substr(x1,2,3)='';</td>
<td></td>
</tr>
<tr>
<td>y1=kupdate(str,2,3);</td>
<td></td>
</tr>
<tr>
<td>z1=kupdates(str,2,3);</td>
<td></td>
</tr>
<tr>
<td>put x1= / y1= / z1=;</td>
<td>x1=1   56</td>
</tr>
<tr>
<td>x2=str; substr(x2,2,3)=&quot;abcd&quot;;</td>
<td></td>
</tr>
<tr>
<td>y2=kupdate(str,2,3,&quot;abcd&quot;);</td>
<td></td>
</tr>
<tr>
<td>z2=kupdates(str,2,3,&quot;abcd&quot;);</td>
<td></td>
</tr>
<tr>
<td>put x2= / y2= / z2=;</td>
<td>x2=1abc56</td>
</tr>
<tr>
<td>x3=str; substr(x3,2,3)=&quot;ab&quot;;</td>
<td></td>
</tr>
<tr>
<td>y3=kupdate(str,2,3,&quot;ab&quot;);</td>
<td></td>
</tr>
<tr>
<td>z3=kupdates(str,2,3,&quot;ab&quot;);</td>
<td></td>
</tr>
<tr>
<td>put x3= / y3= / z3=;</td>
<td>x3=1ab 56</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

Example 2

The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x1=kupdate('感じのテスト',1,2,'漢字');</td>
<td>x1=漢字のテスト</td>
</tr>
<tr>
<td>x2=kupdate(x1,1,2,'kanji');</td>
<td>x2=kaのテスト</td>
</tr>
<tr>
<td>x3=kupdate(x1,1,3);</td>
<td>x3=テスト</td>
</tr>
<tr>
<td>x4=kupdate(x1,1,3,'かんじ');</td>
<td>x4=漢字がんじ</td>
</tr>
<tr>
<td>put x1= / x2= / x3= / x4=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:

- “KUPDATES Function” on page 376
KUPDATEB Function

Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

\[
\text{KUPDATEB}(\text{argument}, \text{position}, n<,\text{characters-to-replace}> ) \\
\text{KUPDATEB}(\text{argument}, \text{position} <,n>, \text{characters-to-replace})
\]

**Required Arguments**

- **argument** specifies a character variable.
- **position** specifies the beginning character position in byte units.
- **n** specifies the length of the substring to be replaced in byte units.

**Restrictions**

- **n** cannot be larger than the length (in bytes) of the expression that remains in **argument** after **position**.
- **n** is optional, but you cannot omit both **n** and **characters-to-replace** from the function.
- **Tip** If you omit **n**, SAS uses all of the characters in **characters-to-replace** to replace the values of **argument**.

- **characters-to-replace** specifies a character expression to replace the contents of **argument**.

**Restriction**

- **characters-to-replace** is optional, but you cannot omit both **characters-to-replace** and **n** from the function.

**Tip** Enclose a literal string of characters in quotation marks.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KUPDATEB function replaces the value of **argument** with the expression in **characters-to-replace**. KUPDATEB replaces **n** byte units starting at the byte unit that you specify in **position**.
If the KUPDATEB function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

**Example**

The following example uses Japanese characters.

```
data _null_;  
x1=kupdateb(’感じのテスト’,1,6,’漢字’);  
x2=kupdateb(x1,1,6,’kanji’);  
x3=kupdateb(x1,1,6);  
x4=kupdateb(x1,7,’かんじ’);  
put x1= / x2= / x3= / x4=;  
run;```

**See Also**

Function:

- “KUPDATE Function” on page 373

---

**KUPDATES Function**

Inserts, deletes, and replaces character value contents.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see *Internationalization Compatibility on page 307*.

**Syntax**

```
KUPDATES(argument,position,n,characters-to-replace)  
KUPDATES(argument,position<n>,characters-to-replace)```

**Required Arguments**

- **argument**
  
  Specifies a character variable.

- **position**
  
  Specifies a numeric expression that is the beginning character position.

**Optional Arguments**

- **n**
  
  Specifies a numeric expression that is the length of the substring to be replaced.
Restrictions

\( n \) cannot be larger than the length of the expression that remains in argument after position.

\( n \) is optional, but you cannot omit both \( n \) and characters-to-replace from the function.

Tip

If you omit \( n \), SAS uses all of the characters in characters-to-replace to replace the values of argument.

**characters-to-replace**
specifies a character expression that replaces the contents of argument.

Restriction

characters-to-replace is optional, but you cannot omit both characters-to-replace and \( n \) from the function.

Tip

Enclose a literal string of characters in quotation marks.

**Details**

See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KUPDATES function replaces the value of argument with the expression in characters-to-replace. KUPDATES replaces \( n \) characters starting at the character that you specify in position.

If the KUPDATES function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 200.

**Examples**

**Example 1**

The following example uses Japanese characters.

```
data null;
x1=kupdates('感じのテスト',1,2,'漢字'); x2=kaのテスト  x3=テスト  x4=漢字かんじ
x2=kupdates(x1,1,2,'kanji');
x3=kupdates(x1,1,3);
x4=kupdates(x1,3,'かんじ');
put x1= / x2= / x3= / x4=;
run;
```

**Example 2**

The following example shows the difference between KUPDATE and KUPDATES.
Statements

```sas
data _null_;
str='123456';
x1=str; substr(x1,2,3)="";
y1=kupdate(str,2,3);
z1=kupdates(str,2,3);
put x1= / y1= / z1=;
x2=str; substr(x2,2,3)="abcd";
y2=kupdate(str,2,3,'abcd');
z2=kupdates(str,2,3,'abcd');
put x2= / y2= / z2=;
x3=str; substr(x3,2,3)="ab";
y3=kupdate(str,2,3,'ab');
z3=kupdates(str,2,3,'ab');
put x3= / y3= / z3=;
run;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1=1 56</td>
<td></td>
</tr>
<tr>
<td>y1=156</td>
<td></td>
</tr>
<tr>
<td>z1=1 56</td>
<td></td>
</tr>
<tr>
<td>x2=1abc56</td>
<td></td>
</tr>
<tr>
<td>y2=1abcd56</td>
<td></td>
</tr>
<tr>
<td>z2=1abc56</td>
<td></td>
</tr>
<tr>
<td>x3=1ab 56</td>
<td></td>
</tr>
<tr>
<td>y3=1ab56</td>
<td></td>
</tr>
<tr>
<td>z3=1ab 56</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Functions:

• “KUPDATE Function” on page 373

KVERIFY Function

Returns the position of the first character that is unique to an expression.

**Category:** DBCS

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Tip:** See “VERIFY Function” in SAS Functions and CALL Routines: Reference

**Syntax**

```sas
KVERIFY(source, excerpt-1, ...excerpt-n)
```

**Required Arguments**

`source`

specifies any SAS character expression.

`excerpt-1, ...excerpt-n`

specifies any SAS character expression.

**Tips**

Enclose a literal string of characters in quotation marks.

If you specify more than one excerpt, separate them with a comma.
Details
See “Internationalization Compatibility for SAS String Functions” on page 307 for restrictions and more information.

The KVERIFY function returns the position of the first character in source that is not present in any excerpt. If KVERIFY finds every character in source in at least one excerpt, it returns a value of 0.

Example
The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td>invalid grade value: 可</td>
</tr>
<tr>
<td>input grade :$2. @@;</td>
<td></td>
</tr>
<tr>
<td>check=&quot;良否&quot;;</td>
<td></td>
</tr>
<tr>
<td>x=verify(grade,check);</td>
<td></td>
</tr>
<tr>
<td>if x gt 0 then put 'Invalid grade value: ' grade;</td>
<td></td>
</tr>
<tr>
<td>cards;</td>
<td></td>
</tr>
<tr>
<td>良 否 良 否 良 否 良 否 可</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

KVERIFYB Function
Returns the position of the first character that is unique to an expression.

<table>
<thead>
<tr>
<th>Category:</th>
<th>DBCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.</td>
</tr>
</tbody>
</table>

Syntax
KVERIFYB(source, excerpt-1,<excerpt-n>)

Required Arguments
source
specifies any SAS character expression.

excerpt
specifies any SAS character expression.

Tip: If you specify more than one excerpt, separate them with a comma.
Details
The KVERIFYB function returns the position of the first character in source that is not present in any excerpt. If KVERIFYB finds every character in source in at least one excerpt, it returns a 0.

Comparisons
KVERIFYB returns byte-based value. KVERIFY returns character-based value. When process SBCS (For example, wlat1 encoding) string, they return identical result. But, in a DBCS session (For example, EUC-JP or SHIFT-JIS encoding), most CJK characters are 2-byte width, that makes these functions return different result.

Example
The following example uses Japanese characters.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>.;</td>
<td>x1=3</td>
</tr>
<tr>
<td>check=’漢字’;</td>
<td>x2=5</td>
</tr>
<tr>
<td>text=’漢字の検索’;</td>
<td></td>
</tr>
<tr>
<td>x1=kverify(text,check);</td>
<td></td>
</tr>
<tr>
<td>put x1=;</td>
<td></td>
</tr>
<tr>
<td>x2=kverifyb(text,check);</td>
<td></td>
</tr>
<tr>
<td>put x2=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATE Function
Converts the SAS date value to the date value of the specified locale by using the date format descriptors.

Category: Date and Time
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Syntax
NLDATE(date,descriptor)

Required Arguments
date
specifies a SAS date value.
descriptor
is a variable or expression that specifies how dates and times are formatted in output. The following descriptors are case sensitive:

#
removes the leading zero from the result.
%% specifies the % character.

%%a specifies the short-weekday descriptor. The range for the day descriptor is Mon–Sun.

%%A specifies the long-weekday descriptor. The range for the long-weekday descriptor is Monday–Sunday.

%%b specifies the short-month descriptor. The range for the short-month descriptor is Jan–Dec.

%%B specifies the long-month descriptor. The range for the long-month descriptor is January–December.

%%C specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.

%%d specifies the day descriptor and uses 0 padding. The range for the day modifier is 01–31.

%%e specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.

%%F specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.

%%j specifies the day-of-year descriptor as a decimal number and uses a leading zero. The range for the day-of-year descriptor is 1–366.

%%m specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.

%%o specifies the month descriptor. The range for the month descriptor is 1–12 with blank padding.

%%u specifies the weekday descriptor as a number in the range 1–7 that represents Monday–Sunday.

%%U specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value using the number of week within the year (Sunday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53 and uses a leading zero and a maximum value of 53.

%%V specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value. The number-of-week value is represented as a decimal number in the range 01–53 and uses a leading zero and a maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both
January 4 and the first Thursday of the year. If the first Monday of January is the
2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding
year.

%w
specifies the weekday descriptor as a number in the range 0–6 that represents
Sunday–Saturday.

%W
specifies the week-number-of-year descriptor by calculating the descriptor value
as SAS date value by using the number of week within the year (Monday is
considered the first day of the week). The number-of-week value is represented
as a decimal number in the range 0–53 and uses a leading zero and a maximum
value of 53.

%y
specifies the year (2-digit) modifier. The range for the year descriptor is 00–99.

%Y
specifies the year (4-digit) descriptor. The range for the year descriptor is 1970–
2069.

Details
The NLDATE function converts the SAS date value to the date value of the specified
locale by using the date descriptors.

Example
The following example shows a log filename that is created from a SAS date value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_Unitedstates;</td>
<td>February-24.log</td>
</tr>
<tr>
<td>logfile=nldate('24Feb2003'd,'%B-%d.log');</td>
<td></td>
</tr>
<tr>
<td>put logfile;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>Februar-24.log</td>
</tr>
<tr>
<td>logfile=nldate('24Feb2003'd,'%B-%d.log');</td>
<td></td>
</tr>
<tr>
<td>put logfile;</td>
<td></td>
</tr>
</tbody>
</table>

The following example shows a weekday name that is created from a SAS date value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_unitedstates;</td>
<td>Monday</td>
</tr>
<tr>
<td>weekname=nldate('24Feb2003'd,'%A');</td>
<td></td>
</tr>
<tr>
<td>put weekname;</td>
<td></td>
</tr>
<tr>
<td>Statements</td>
<td>Results</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td><code>options locale=German_Germany;</code></td>
<td>Montag</td>
</tr>
<tr>
<td><code>weekname=nldate('24Feb2003'd,'%A');</code></td>
<td></td>
</tr>
<tr>
<td><code>put weekname;</code></td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Format:**
- “NLDATEm. Format” on page 130

### NLDATM Function

Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors.

**Category:** Date and Time

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

### Syntax

**NLDATM**(`datetime,descriptor`)  

**Required Arguments**

- **datetime**
  - specifies a SAS datetime value.

- **descriptor**
  - is a variable or expression that specifies how dates and times are formatted in output.
  - The following descriptors are case sensitive:
    - `#`
      - removes the leading zero from the result.
    - `%%`
      - specifies the % character.
    - `%a`
      - specifies the short-weekday descriptor. The range for the day descriptor is Mon–Sun.
    - `%A`
      - specifies the long-weekday descriptor. The range for the long-weekday descriptor is Monday–Sunday.
    - `%b`
      - specifies the short-month descriptor. The range for the short-month descriptor is Jan–Dec.
%B
specifies the long-month descriptor. The range for the long-month descriptor is January–December.

%C
specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.

%D
specifies the day descriptor and uses 0 padding. The range for the day descriptor is 01–31.

%E
specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.

%F
specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.

%H
specifies the hour descriptor that is based on a 24-hour clock. The range for the hour descriptor is 00–23.

%I
specifies the hour descriptor that is based on a 12-hour clock. The range for the hour descriptor is 01–12.

%J
specifies the day-of-year descriptor as a decimal number and uses a leading zero. The range for the day-of-year descriptor is 1–366.

%M
specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.

%M
specifies the minute descriptor. The range for the minute descriptor is 00–59.

%O
specifies the month descriptor and uses blank padding. The range for the month descriptor is 1–12.

%p
specifies a.m. or p.m. descriptor.

%S
specifies the second descriptor. The range for the second descriptor is 00–59.

%U
specifies the weekday descriptor as a number in the range of 1–7 that represents Monday–Sunday.

%V
specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value and uses the number-of-week value within the year (Sunday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53. A leading zero and a maximum value of 53 is used.

%V
specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value. The number-of-week value is represented as a decimal
number in the range 01–53. A leading zero and a maximum value of 53 is used. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4 and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year.

%w
specifies the weekday descriptor as a number in the range of 0–6 that represents Sunday–Saturday.

%W
specifies the week-number-of-year descriptor by calculating the descriptor value as SAS date value using the number of week within the year (Monday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range of 0–53. A leading zero and a maximum value of 53 are used.

%y
specifies the year (2-digit) descriptor. The range for the year descriptor is 00–99.

%Y
specifies the year (4-digit) descriptor. The range for the year descriptor is 1970–2069.

Details
The NLDATM function converts the SAS datetime value to the datetime value of the specified locale by using the datetime descriptors.

Example
The following example shows a time (a.m or p.m.) that is created from a SAS datetime value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options locale=English;</code></td>
<td></td>
</tr>
<tr>
<td><code>time_ampm=nldatm('24Feb2003:12:39:43''dt','%I%p');</code></td>
<td></td>
</tr>
<tr>
<td><code>put time_ampm;</code></td>
<td>12PM</td>
</tr>
<tr>
<td><code>options locale=German;</code></td>
<td></td>
</tr>
<tr>
<td><code>time_ampm=nldatm('24Feb2003:12:39:43''dt','%I%p');</code></td>
<td></td>
</tr>
<tr>
<td><code>put time_ampm;</code></td>
<td>12nachm</td>
</tr>
</tbody>
</table>

See Also
Format:
- “NLDATMw. Format” on page 151
**NLTIME Function**

Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors.

**Category:** Date and Time

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

```
NLTIME(time|datetime,descriptor,startpos)
```

**Required Arguments**

- **time** specifies a SAS time value.
- **datetime** specifies a SAS datetime value.
- **descriptor** is a variable, or expression, that specifies the value of a descriptor. You can enter the following descriptors in uppercase or lowercase:
  - `%#` removes the leading zero from the result.
  - `%%` specifies the % character.
  - `%H` specifies the hour descriptor that is based on a 24-hour clock. The range for the hour descriptor is 00–23.
  - `%I` specifies the hour descriptor that is based on a 12-hour clock. The range for the hour descriptor is 01–12.
  - `%M` specifies the minute modifier. The range for the minute descriptor is 00–59.
  - `%P` specifies the a.m. or p.m. descriptor.
  - `%S` specifies the second descriptor. The range for the second descriptor is 00–59.
- **startpos** is an integer that specifies the position at which the search should start and that specifies the direction of the search.

**Details**

The NLTIME function converts a SAS time or datetime value to the time value of the specified locale by using the time descriptors.
Example

The following example shows an a.m. or p.m. time that is created from a SAS time.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English;</td>
<td></td>
</tr>
<tr>
<td>time_ampm=nltime('12:39:43't,'%i%p');</td>
<td></td>
</tr>
<tr>
<td>put time_ampm;</td>
<td>00 PM</td>
</tr>
<tr>
<td>options locale=German;</td>
<td></td>
</tr>
<tr>
<td>time_ampm=nltime('12:39:43't,'%i%p');</td>
<td></td>
</tr>
<tr>
<td>put time_ampm;</td>
<td>00 nachm</td>
</tr>
</tbody>
</table>

See Also

Format:
- “NLTIMEw. Format” on page 265

SASMSG Function

Specifies a message from a data set. The returned message is based on the current locale and a specified key.

**Category:** Locale

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

SASMSG *(BASENAME", "KEY", <"QUOTE"|"DQUOTE"|"NOQUOTE">*<, "substitution 1", ..., "substitution 7">*)

**Required Arguments**

*BASENAME*

the name of the data set where the message is located.

*KEY*

the message key.

*Note:* If you specify an invalid key name, then the key name is returned.

*QUOTE|DQUOTE|NOQUOTE*

specifies the type of quotation marks that are added to the message text and substitution strings.
Default DQUOTE

string substitutions. The maximum string substitutions is 7.

Details

The SAS message data set must be a 7-bit ASCII data set. Any character that cannot be represented in the 7-bit ASCII encoding is represented in the Unicode escape format of \uxxxx, where \uxxxx is the base 10 numeric representation of the Unicode value of the character.

The data set used by the SASMSG function must have been created specifically for use with this function. The data set must contain the following variables:

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>locale</td>
<td>char</td>
<td>5</td>
<td>language of the message</td>
</tr>
<tr>
<td>2</td>
<td>key</td>
<td>char</td>
<td>60</td>
<td>key to identify the message</td>
</tr>
<tr>
<td>3</td>
<td>lineno</td>
<td>num</td>
<td>5</td>
<td>line # of the message in reverse order</td>
</tr>
<tr>
<td>4</td>
<td>text</td>
<td>text</td>
<td>1,200</td>
<td>text of the message</td>
</tr>
</tbody>
</table>

The data set must be sorted on the following variables: locale, key, and lineno. The variable lineno must be in descending order. A composite index on locale and key must be defined. Here is a sample program to sort and create an indexed data set:

```sas
%let basename=MyProduct;

proc sort data=t.&basename;
  by locale key descending lineno;
run;

proc datasets lib=t
  memtype=data;
  modify &basename;
  index create indx=(LOCALE KEY);
run;
quit;
```

The returned message is based on the LOCALE system option. The LOCALE option is represented by \ll_RR where \ll represents the two-letter language code and RR represents the two-letter region code. If a match is not found, then the function searches for a match with the language only. If the pair locale and key are still not found, then the function defaults to the English language (en). If the key does not exist for English (en), then the key name is returned.

You can alter formatting. You can use string substitution by using the format code %s. You can change the order of substitution. In some cases, translation of a message to a language other than English might require changing the order of substitutions. You can change the order by placing an argument number specification, #nn, within a format.
string, where \( nn \) is the number of the argument in the substitution list. The following example demonstrates the order:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg = sasmsg(&quot;nls.mymsg&quot;,&quot;IN_CD_LOG&quot;,&quot;noquote&quot;,&quot;cat&quot;,&quot;dog&quot;);</code></td>
<td></td>
</tr>
<tr>
<td><code>IN_CD_LOGINFO = My %#1s. Your %#2s</code></td>
<td></td>
</tr>
<tr>
<td><code>IN_CD_LOGINFO = My %#2s. Your %#1s</code></td>
<td><code>msg= My cat. Your dog.</code></td>
</tr>
<tr>
<td><code>options locale = es_ES;</code></td>
<td><code>msg= My dog. Your cat.</code></td>
</tr>
</tbody>
</table>

The SASMSG function can be used in the open code macro with the `%SYSFUNC` macro function.

Arguments that are passed to a function called by the `%SYSFUNC` macro must not be in quotation marks. Arguments passed to the SASMSG function outside of `%SYSFUNC` must be quoted.

When the SASMSG function is used with the `%SYSFUNC` macro function, the returned string is wrapped with the `%NRBQUOTE` function.

## Examples

### Example 1

The following example demonstrates the formatting feature of SASMSG:

```sas
%macro demo_sasmsg;
  data _null_;
  msg = sasmsg("nls.mymsg","IN_APW_SAVE_OK","noquote");
  put msg=;
  run;
%mend demo_sasmsg;
```

<table>
<thead>
<tr>
<th>SAS Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options locale = en_US;</code></td>
<td><code>msg= The Access Control key was successfully saved.</code></td>
</tr>
<tr>
<td><code>options locale = es_ES;</code></td>
<td><code>msg= La clave de control de acceso se ha guardado.</code></td>
</tr>
<tr>
<td><code>options locale = french_France;</code></td>
<td><code>msg= La clé de contrôle d’accès a bien été enregistrée.</code></td>
</tr>
</tbody>
</table>

### Example 2

The following example demonstrates the open macro feature:

```sas
%MACRO PRT(loc, tb, key);
  option locale=&loc;
  %PUT %SYSFUNC(SASMSG(&tb,&key) );
%mEND PRT;
```
SASMSGL Function

Specifies a message from a data set. The message is based on a specified locale value and a specified key value.

Syntax

SASMSGL("basename", "key", "locale", (, , <"q"|"d"|"n"> <, "substitution 1", ..., "substitution 6">))

Required Arguments

**basename**
the name of the data set where the message is located.

**key**
the message key.

*Note:* If you specify an invalid key name, then the key name is returned.

**locale**
the posix locale value (ll RR).

**quote|dquote|noquote**
specifies the type of quotation marks that are added to the message text and substitution strings.

*Default* DQUOTE

**substitution**
string substitutions. The maximum string substitutions is 6.

Details

The SAS message data set must be a 7-bit ASCII data set. Any character that cannot be represented in the 7-bit ASCII encoding is represented in the Unicode escape format of "\uxxxx", where the xxxx is the base 10 numeric representation of the Unicode value of the character.

The data set used by SASMSGL function must have been created specifically for use with this function. The data set must contain the following variables:
<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>locale</td>
<td>char</td>
<td>5</td>
<td>language of the message</td>
</tr>
<tr>
<td>2</td>
<td>key</td>
<td>char</td>
<td>60</td>
<td>key to identify the message</td>
</tr>
<tr>
<td>3</td>
<td>lineno</td>
<td>num</td>
<td>5</td>
<td>line number of the message in reverse order</td>
</tr>
<tr>
<td>4</td>
<td>text</td>
<td>text</td>
<td>1200</td>
<td>text of the message</td>
</tr>
</tbody>
</table>

The data set must be sorted on the following variables: locale, key, and lineno. The variable lineno must be in descending order. A composite index on locale and key must be defined. Here is a sample program to sort and create an indexed data set:

```sas
%let basename=MyProduct;

proc sort data=t.&basename;
by locale key descending lineno;
run;

proc datasets lib=t
memtype=data;
modify &basename;
index create indx=(LOCALE KEY);
run;
quit;
```

The returned message is based on the LOCALE system option. The LOCALE option is represented by ll_RR where ll represents the two-letter language code and RR represents the two-letter region code. If a match is not found, then the function searches for a match with the language only. If the pair locale and key are still not found, then the function defaults to the English language (en). If the key does not exist for English (en), then the key name is returned.

You can alter formatting. You can use string substitution by using the format code %s. You can change the order of substitution. In some cases, translation of a message to a language other than English might require changing the order of substitutions. You can change the order by placing an argument number specification, #nn, within a format string, where nn is the number of the argument in the substitution list. The following example demonstrates changing the order:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg = sasmsgl(&quot;nls.mymag&quot;,&quot;IN_CD_LOG&quot;,&quot;en_US&quot;,&quot;N&quot;, &quot;cat&quot;,&quot;dog&quot;);</code></td>
<td>IN_CD_LOGINFO = My %#1s. Your %#2s</td>
</tr>
<tr>
<td><code>msg = sasmsgl(&quot;nls.mymag&quot;,&quot;IN_CD_LOG&quot;,&quot;en_US&quot;,&quot;N&quot;, &quot;cat&quot;,&quot;dog&quot;);</code></td>
<td>IN_CD_LOGINFO = My %#2s. Your %#1s</td>
</tr>
</tbody>
</table>

The returned message is based on the LOCALE system option. The LOCALE option is represented by ll_RR where ll represents the two-letter language code and RR represents the two-letter region code. If a match is not found, then the function searches for a match with the language only. If the pair locale and key are still not found, then the function defaults to the English language (en). If the key does not exist for English (en), then the key name is returned.

You can alter formatting. You can use string substitution by using the format code %s. You can change the order of substitution. In some cases, translation of a message to a language other than English might require changing the order of substitutions. You can change the order by placing an argument number specification, #nn, within a format string, where nn is the number of the argument in the substitution list. The following example demonstrates changing the order:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg = sasmsgl(&quot;nls.mymag&quot;,&quot;IN_CD_LOG&quot;,&quot;en_US&quot;,&quot;N&quot;, &quot;cat&quot;,&quot;dog&quot;);</code></td>
<td>IN_CD_LOGINFO = My %#1s. Your %#2s</td>
</tr>
<tr>
<td><code>msg = sasmsgl(&quot;nls.mymag&quot;,&quot;IN_CD_LOG&quot;,&quot;en_US&quot;,&quot;N&quot;, &quot;cat&quot;,&quot;dog&quot;);</code></td>
<td>IN_CD_LOGINFO = My %#2s. Your %#1s</td>
</tr>
</tbody>
</table>
The SASMSGL function can be used in the open code macro with the %SYSFUNC macro function.

Arguments that are passed to a function called by the %SYSFUNC macro must not be in quotation marks. Arguments passed to the SASMSGL function outside of %SYSFUNC must be quoted.

When the SASMSGL function is used with the %SYSFUNC macro function, the returned string is wrapped with the %NRBQUOTE function.

Examples

**Example 1**
The following example demonstrates the formatting feature of SASMSGL:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>sasmsgl(“nls.mymsg”, “IN_APW_SAVE_OK”, “en_US”, “n”)</td>
<td>“The Access Control key was successfully saved.”</td>
</tr>
<tr>
<td>sasmsgl(“nls.mymsg”, “IN_APW_SAVE_OK”, “es_ES”, “n”)</td>
<td>“La clave de control de acceso se ha guardado.”</td>
</tr>
<tr>
<td>sasmsgl(“nls.mymsg”, “IN_APW_SAVE_OK”, “fr”, “n”)</td>
<td>“La clé de contrôle d’accès a bien été enregistrée.”</td>
</tr>
</tbody>
</table>

**Example 2**
The following example demonstrates the open macro feature:

<table>
<thead>
<tr>
<th>SAS Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>%PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, en_US));</td>
<td>“Edit”</td>
</tr>
<tr>
<td>%PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, es_ES));</td>
<td>“Editar”</td>
</tr>
<tr>
<td>%PUT %SYSFUNC(SASMSGL(NLS.MYDS, IN_ASD_LABEL, fr));</td>
<td>“Modifier”</td>
</tr>
</tbody>
</table>

**SORTKEY Function**

Creates a linguistic sort key.

**Category:** Locale

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

sortKey(string, <locale, strength, case_order, numeric_order, >)
**Required Arguments**

**string**
character expression

**locale**
specifies the locale name in the form of a POSIX name (ja_JP). See Table 20.1 on page 697 for a list of locale names and POSIX values.

**strength**
The value of strength is related to the collation level. There are five collation-level values. The following table provides information about the five levels. The default value for strength is related to the locale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Type of Collation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY or P</td>
<td>PRIMARY specifies differences between base characters (for example, &quot;a&quot; &lt; &quot;b&quot;).</td>
<td>It is the strongest difference. For example, dictionaries are divided into different sections by base character.</td>
</tr>
<tr>
<td>SECONDARY or S</td>
<td>Accents in the characters are considered secondary differences (for example, &quot;as&quot; &lt; &quot;äs&quot; &lt; &quot;at&quot;).</td>
<td>Other differences between letters can also be considered secondary differences, depending on the language. A secondary difference is ignored when there is a primary difference anywhere in the strings.</td>
</tr>
<tr>
<td>TERTIARY or T</td>
<td>Upper and lowercase differences in characters are distinguished at the tertiary level (for example, &quot;ao&quot; &lt; &quot;Ao&quot; &lt; &quot;aö&quot;).</td>
<td>An example is the difference between large and small Kana. A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings.</td>
</tr>
<tr>
<td>QUATERNARY or Q</td>
<td>When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, &quot;ab&quot; &lt; &quot;a-b&quot; &lt; &quot;aB&quot;).</td>
<td>This difference is ignored when there is a primary, secondary, or tertiary difference. The quaternary level should be used if ignoring punctuation is required or when processing Japanese text.</td>
</tr>
<tr>
<td>IDENTICAL or I</td>
<td>When all other levels are equal, the identical level is used as a tiebreaker. The Unicode code point values of the NFD form of each string are compared at this level, just in case there is no difference at levels 1-4.</td>
<td>For example, only Hebrew cantillation marks are distinguished at this level. This level should be used sparingly, as only code point values differences between two strings is an extremely rare occurrence.</td>
</tr>
</tbody>
</table>

**case order**
sorts uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL. The following table provides the values and information for the case order argument.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER or U</td>
<td>Sorts uppercase letters first, then the lowercase letters.</td>
</tr>
<tr>
<td>LOWER or L</td>
<td>Sorts lowercase letters first, then the uppercase letters.</td>
</tr>
</tbody>
</table>

**numeric order**

orders numbers by the numeric value instead of the number's characters.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC or N</td>
<td>Order numbers (integers) by the numeric value. For example, &quot;8 Main St.&quot; would sort before &quot;45 Main St.&quot;.</td>
</tr>
</tbody>
</table>

**collation order**

There are two types of collation values: Phonebook and Traditional. If you do not select a collation value, then the user's locale-default collation is selected. The following table provides more information.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHONEBOOK or P</td>
<td>specifies a phonebook style ordering of characters. Select PHONEBOOK only with the German language.</td>
</tr>
<tr>
<td>TRADITIONAL or T</td>
<td>specifies a traditional style ordering of characters. Select TRADITIONAL only with the Spanish language.</td>
</tr>
</tbody>
</table>

**Details**

The SORTKEY function creates a linguistic sort key for data. You must enter at least one argument. If the length of the variable that receives the key is not large enough, the data truncates, and a warning is displayed.

locale

Locale values use the POSIX name (ll_RR). LL represents the two-letter language code, and RR represents the two-letter region code. For example, en_US is the POSIX name for English, United States. en represents the English language, and US represents the United States. If a locale value is not specified, then the session locale is used.

strength

The strength argument determines whether accents or case affect collating or matching text. If no value is specified for strength, then the locale determines the value. The following values can be specified for strength.

- PRIMARY
  This value includes base letters. An example is the letters, A, a, and Å are all processed the same.

- SECONDARY
  This value processes data the same as PRIMARY, and accents are processed. The letters A and a are processed equally, and Å is processed as an accented character.
TERTIARY
This value processes data the same as SECONDARY, and the character's case is processed. For example, A, a, and Å are all processed differently.

QUATERNARY
This value processes data the same as TERTIARY, and punctuation is processed.

IDENTICAL
This value process data the same as QUATERNARY, and code point is processed.

case order
specifies to sort data using uppercase or lowercase letter. The following table shows examples of specifying the UPPER value or the LOWER value.

<table>
<thead>
<tr>
<th>UPPER</th>
<th>LOWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aztec</td>
<td>aztec</td>
</tr>
<tr>
<td>aztec</td>
<td>Aztec</td>
</tr>
<tr>
<td>Mars</td>
<td>mars</td>
</tr>
<tr>
<td>mars</td>
<td>Mars</td>
</tr>
</tbody>
</table>

collation order
The collation order value PHONEBOOK is ignored unless the locale is a German language.

The collation order value TRADITIONAL is ignored unless the locale is a Spanish language.

A warning message is displayed for other locales.

---

**SETLOCALE Function**

Specifies the locale keys for the current SAS locale.

- **Category:** Locale
- **Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

Setting SAS Locale

```
SETLOCALE (sas_locale)
```

Customize single locale elements

```
SETLOCALE (key,value)
```

Customize single locale elements

```
SETLOCALE (category_name, sas_locale)
```
**Required Arguments**

*sas_locale*

specifies a SAS locale name by using the SAS name or the posix name. You can also specify the locale alias.

*key*

specifies a SAS locale element key. See the list of element keys in the Details section.

*value*

specifies a value for the locale element.

*category_name*

specifies the category name:

- LC_TIME
- LC_MONETARY
- LC_NUMERIC
- LC_ALL

**Details**

You can modify the following locale elements. The value of key must be less than the value of max length. You can specify the following values for type:

- 0  String.
- 1  Unsigned integer. You must use double quotation marks.

<table>
<thead>
<tr>
<th>Locale Element Key</th>
<th>Max Length</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATESTYLE</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PAPERSIZE</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTITLE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTEXT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMFONT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SORTSEQ</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>MESSAGES</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATETIME</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_TIME</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>------</td>
<td>---------------</td>
</tr>
<tr>
<td>FORMATNAME_NUMERIC</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_PERCENT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SANS_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_CURSIVE</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_FANTASY</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_MONOSPACED</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BRUSH</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SWISS</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ITALIC</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DATE_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_AMPM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME_AMPM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>TIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_WEEK_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_WEEK_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYMM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYMM_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_MMDD_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>ABMON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>DAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>AM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>PM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>INT_CURRENCY_SYMBOL</td>
<td>3</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>CURRENCY_SYMBOL</td>
<td>32</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_DECIMAL_POINT</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_THOUSANDS_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>MON_NEGATIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_INT_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_SIGN_POSITION</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_SIGN_POSITION</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>DECIMAL_POINT</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>THOUSANDS_SEP</td>
<td>1</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>NEGATIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>P_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>N_SIGN_POSN</td>
<td>3</td>
<td>1</td>
<td>LC_NUMERIC</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WIDTH</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Examples

Example 1
In the following locale example, the SETLOCALE function specifies the locale Japanese (ja_JP). The SETLOCALE function returns the previous locale. In this example, the previous locale was English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x=setlocale('ja_JP');</td>
<td></td>
</tr>
<tr>
<td>put x=; run;</td>
<td></td>
</tr>
<tr>
<td>x=English_UnitedStates</td>
<td></td>
</tr>
</tbody>
</table>

Example 2
In the following example, the SETLOCALE function returns the locale name where the element values are being changed:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>x=setlocale(&quot;LC_MONETARY&quot;, 'zh_CN');</td>
<td></td>
</tr>
<tr>
<td>put x=; run;</td>
<td></td>
</tr>
<tr>
<td>x=Japanese_Japan</td>
<td></td>
</tr>
</tbody>
</table>

Example 3
In the following example, the SETLOCALE function changes the value of the specified key, DATE_YEAR_FORMAT:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data _null;</td>
<td></td>
</tr>
<tr>
<td>x=setlocale('DATE_YEAR_FORMAT', '¥%Y');</td>
<td></td>
</tr>
<tr>
<td>put x=; run;</td>
<td></td>
</tr>
<tr>
<td>x=%Y¥</td>
<td></td>
</tr>
</tbody>
</table>

TZONEID Function
Returns the current time zone ID.

Category: Date and Time
Alias: TZID
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.
Syntax
TZONEID<time-zone-id>

Optional Argument
time-zone-id
specifies a region or area value that is defined by SAS. When you specify a zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details
The TZONEID function returns a blank value if the TIMEZONE= option is blank or a user-defined time zone is specified.

The TZONEID function validates the timezone ID. If you specify the timezone ID, the function returns the timezone ID if it is valid or returns a blank value if the ID is invalid.

Example
In the first example, the TIMEZONE option is set to JST. In the second example, TIMEZONE is set to a blank value. In the third example TIMEZONE is set to user-specified time zone. In the fourth example a valid timezoneid and an invalid timezoneid is displayed.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone=jst;</td>
<td>tzid=ASIA/TOKYO</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzid=tzoneid() ;</td>
<td></td>
</tr>
<tr>
<td>put tzid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options timezone='';</td>
<td>tzid=</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzid=tzoneid() ;</td>
<td></td>
</tr>
<tr>
<td>put tzid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options timezone='xxx-12';</td>
<td>tzid=</td>
</tr>
<tr>
<td>/* user defined timezone */</td>
<td></td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>tzid=tzoneid() ;</td>
<td></td>
</tr>
<tr>
<td>put tzid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>data null;</td>
<td>name_valid=ASIA/TOKYO</td>
</tr>
<tr>
<td>name_valid=tzoneid('asia/tokyo');</td>
<td></td>
</tr>
<tr>
<td>name_invalid=tzoneid('Milky Way');</td>
<td></td>
</tr>
<tr>
<td>put name_valid =;</td>
<td></td>
</tr>
<tr>
<td>put name_invalid=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
**TZONENAME Function**

Returns the current standard or daylight savings time, time zone name.

**Category:** Date and Time  
**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

```
TZONENAME()
TZONENAME<time-zone-id,datetime>
```

**Optional Arguments**

- `time-zone-id` specifies a region/area value that is defined by SAS. When you specify a zone ID, the time zone that SAS uses is determined by time zone name and daylight savings time rules.

See For a list of the time zone IDs, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819.

- `datetime` specifies a SAS datetime value.

**Details**

The TZONENAME function returns a blank value if the TIMEZONE= option is blank.

The TZONENAME function returns the timezone name based on the specified timezone and datetime. If the SAS datetime is not specified, then the current date is used. If TZID is not specified, then the timezone ID that is specified with the TIMEZONE= option is used.

**Example**

In the first example, the TIMEZONE option is set to a blank value. In the second example, TIMEZONE is set to timezone name, JST. In the third example, TIMEZONE is set to a user-specified time zone. In the fourth example, TIMEZONE is set to a time zone ID.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options tz=''; data <em>null</em>; tzone=tzonenumber(); put tzone =; run;</td>
<td>tzone=</td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
`options tz='jst';` | `tzname=JST`
`data _null_;` | `tzname=tzonename();`
`put tzname =;` | `run;`

`options tz='xxx-12';` | `tzname=XXX`
`data _null_;` | `tzname=tzonename();`
`put tzname =;` | `run;`

`options tz='American/Chicago';` | `tzname=CDT`
`data_null_;` | `tzname=tzonename('01SEP2014:01:01:01'dt);`
`put tzname =;` | `run;`

TZONEOFF Function

Returns the user time zone offset.

**Category:** Date and Time

**Restriction:**
This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

`TZONEOFF()`

`TZONEOFF(<time-zone-id, datetime>)`

**Optional Arguments**

**time-zone-id**

specifies a region/area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by time zone name and daylight savings time rules.

**See** For a list of time zone IDs, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819.

**datetime**

specifies a SAS datetime value.

**Details**

If no arguments are specified, the TZONEOFF function returns the time zone offset for the specified TIMEZONE option. The TZONEOFF (time-zone-id) function with the time zone ID argument returns the time zone offset for the specified time zone ID. The
TZONEOFF function with the time zone ID argument returns the time zone offset for the specified time zone name. If the time zone name is not valid for the current locale, you receive an error. We recommend that you use the time zone ID, since it is not locale dependent.

If SASDTM is not provided, TZONEOFF returns the current timezone offset. If SASDTM is provided, it returns the offset to get the local time for specified time value.

Example

The first example has no argument, so the TZONEOFF function returns an offset for the current SAS session. The second example returns an offset based on a specific time zone ID. The third example returns an offset based on a specific time zone ID and a specific date and time. The fourth example returns an offset based on the Time Zone option and a specific date.

If the SAS datetime is not specified, then the TZONEOFF function returns the current timezone offset. If the SAS datetime is specified, then the function returns the offset to provide the local time for the specified time value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option TIMEZONE='AUSTRALIA/MELBOURNE'; %PUT %SYSFUNC(TZONEOFF());</td>
<td>39600</td>
</tr>
<tr>
<td>option TIMEZONE='AUSTRALIA/MELBOURNE'; %PUT %SYSFUNC(TZONEOFF('EUROPE/ROME'));</td>
<td>3600</td>
</tr>
<tr>
<td>data <em>null</em>; dt1='05DEC2012:08:17:52'dt; dt2='05JUN2012:08:17:52'dt; offset1= TZONEOFF('EUROPE/MOSCOW', dt1); offset2= TZONEOFF('EUROPE/MOSCOW', dt2); put offset1= / offset2= ; run ;</td>
<td>offset1=10800 offset2=14400</td>
</tr>
<tr>
<td>option TIMEZONE='EUROPE/MOSCOW' ; data <em>null</em>; dt1='05DEC2012:08:17:52'dt; dt2='05JUN2012:08:17:52'dt; offset1= TZONEOFF(dt1); offset2= TZONEOFF(dt2); put offset1= / offset2= ; run ;</td>
<td>offset1=10800 offset2=14400</td>
</tr>
<tr>
<td>option TIMEZONE='EUROPE/MOSCOW' ; data <em>null</em>;</td>
<td></td>
</tr>
</tbody>
</table>

TZONES2U Function

Converts a SAS date time value to a UTC date time value.

Category: Date and Time

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.
Syntax
TZONES2U <datetime, time-zone-id>

Optional Arguments
datetime
specifies a SAS datetime value.
time-zone-id
specifies a region or area value that is defined by SAS. When you specify a time
zone ID, the time zone that SAS uses is determined by time zone name and daylight
savings time rules.

See  For a list of time zone IDs, see Appendix 2, “Time Zone IDs and Time Zone
Names,” on page 819

Details
The TZONES2U() function returns UTC-based time for the specified TIMEZONE. The
TZONES2U(time-zone-id) function with the time zone ID argument returns UTC-based
time for the specified time zone ID. If the time zone name is not valid for the current
locale, you receive an error.

Example
The following example converts a SAS date time into UTC time.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=ja_JP TZ='JST' ;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>data <em>null</em> ;</td>
<td></td>
</tr>
<tr>
<td>dt='05Nov2012:08:17:52'dt</td>
<td>utc1=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>utc1 = tzones2u(dt) ;</td>
<td></td>
</tr>
<tr>
<td>utc2 = tzones2u(dt,'ASIA/TOKYO') ;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>utc3 = tzones2u(dt,'JST') ;</td>
<td>utc2=2012-11-04T23:17:52+00:00</td>
</tr>
<tr>
<td>put dt= /utc1= is8601dz. // ;</td>
<td></td>
</tr>
<tr>
<td>put dt= /utc2= is8601dz. // ;</td>
<td></td>
</tr>
<tr>
<td>put dt= /utc3= is8601dz. // ;</td>
<td>dt=1667722672</td>
</tr>
<tr>
<td>run ;</td>
<td>utc3=2012-11-04T23:17:52+00:00</td>
</tr>
</tbody>
</table>

TZONEDSTNAME Function

Returns a daylight savings time name.

Category: Date and Time

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS,
DBCS, and MBCS (UTF8). For more information, see Internationalization
Compatibility on page 307.
Syntax
TZONEDSTNAME()
TZONEDSTNAME<time-zone-id>

Optional Argument

time-zone-id
specifies a region or area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details

If the TZID is not specified, then the TIMEZONE ID that is specified by the TIMEZONE= option is used. If the daylight savings time is not specified, then the function returns a blank name.

Example

The following example specifies the America/Chicago time zone.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone='America/Chicago';</td>
<td>dstname=CDT</td>
</tr>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>dstname=tzonedstname();</td>
<td></td>
</tr>
<tr>
<td>put dstname;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

TZONEDSTOFF Function

Returns the time zone offset value for the specified daylight savings time.

Category: Date and Time
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Syntax
TZONEDSTOFF()
TZONEDSTOFF<time-zone id>

Optional Argument

time zone id
specifies a region or area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.
Details
If the TZID is not specified, then the TIMEZONE ID that is specified by the
TIMEZONE= option is used. If the daylight savings time is not specified, then the
function returns a blank name.

Example
The following example specifies the America/Chicago time zone.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone='America/Chicago';</td>
<td>dstoff=18000</td>
</tr>
<tr>
<td>data null;</td>
<td></td>
</tr>
<tr>
<td>dstoff=tzonedstoff();</td>
<td></td>
</tr>
<tr>
<td>put dstoff;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

TZONESTTNAME Function
Returns a standard time zone name.

Category: Date and Time
Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

Syntax
TZONESTTNAME()

Optional Argument
time-zone-id

Details
If the TZID is not specified, then the TIMEZONE ID that is specified by the
TIMEZONE= option is used.

Example
The following example specifies the Asia/Osaka time zone.
**Statements**

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone='Asia/Osaka'; data null; name=tzonesttname(); put name; run;</td>
<td>JST</td>
</tr>
</tbody>
</table>

**TZONESTTOFF Function**

Returns the time zone offset value for the specified standard time.

**Category:** Date and Time  
**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

TZONESTTOFF()

TZONESTTOFF<time-zone id>

**Optional Argument**

**time-zone-id**  
specifies a region or area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

**Details**

If the TZID is not specified, then the TIMEZONE ID that is specified by the TIMEZONE= option is used. If standard time is not specified, then the function returns a blank name.

**Example**

The following example specifies the Asia/Osaka time zone.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options timezone='Asia/Osaka'; data null; name=tzonestttoff{}; put name; run;</td>
<td>32400</td>
</tr>
</tbody>
</table>
TZONEU2S Function

Converts a UTC date time value to a SAS date time value.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.</td>
</tr>
</tbody>
</table>

Syntax

TZONEU2S <UTC date time value, time-zone id>

Optional Arguments

UTC date time value
- specifies a Coordinated Universal Time (UTC) datetime value

time-zone id
- specifies a region or area value that is defined by SAS. When you specify a zone ID, the time zone that SAS uses is determined by the time zone name and daylight savings time rules.

Details

The TZONEU2S(datetime) function returns the SAS datetime for a UTC time for the specified TIMEZONE option. The TZONEU2S(datetime, time-zone-id) function with the time zone ID argument, returns the SAS datetime for the UTC time for the specified time zone ID. If the time zone name is not valid for the current locale, you receive an error.

Example

The following example converts a UTC date time to three specific SAS date time values.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=fr_FR TZ='AMERICA/DENVER'; data <em>null</em>; utc_date = '2012-09-02T02:34:56+00:00'; udt = input(utc_date, is8601dz.); sdt1 = tzoneu2s(udt); sdt2 = tzoneu2s(udt, 'EUROPE/AMSTERDAM'); sdt3 = tzoneu2s(udt, 'CET'); put sdt1= datetime. / sdt2= datetime. / sdt3= datetime.; run;</td>
<td>sdt1=01SEP12:20:34:56 sdt2=02SEP12:03:34:56 sdt3=02SEP12:03:34:56</td>
</tr>
</tbody>
</table>
TRANTAB Function

Transcodes data by using the specified translation table.

**Category:** Character

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

### Syntax

```
TRANTAB(string, trantab_name)
```

### Required Arguments

- **string**: input data that is transcoded.
- **trantab_name**: translation table.

### Details

The TRANTAB function transcodes a data string by using a translation table to remap the characters from one internal representation to another. The encoding of the data in the input string must match the encoding of table 1 in the translation table. The TRANTAB function remaps the data from the encoding using table 1.

Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= and ENCODING= system options as an improvement on direct use of translation tables. In SAS 9 translation tables are used only for transcoding external files. There is direct transcoding in SAS files, between the session encodings. SAS 9.2 supports the TRANTAB function for backward compatibility.

The LOCALE= system option is preferred in later SAS releases.

**CAUTION:**

Only experienced SAS users should use the TRANTAB function.

### Example

The following example uses a translation table that transcodes data that is encoded in Latin2 to an uppercase Latin2 encoding:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>teststrg=trantab('testing','lat2_ucs');</code></td>
<td>Testing</td>
</tr>
<tr>
<td><code>put teststrg;</code></td>
<td></td>
</tr>
</tbody>
</table>
**UNICODE Function**

Converts Unicode characters to the current SAS session encoding.

**Category:** Character

**Restriction:** This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.

**Syntax**

\[
\text{STR=} \text{UNICODE}(<\text{instr}>, (<\text{Unicode type}>) )
\]

**Required Arguments**

*str*
Data string that has been converted to the current SAS session encoding.

*instr*
input data string.

*Unicode type*
Unicode character formats

- **ESC**  Unicode Escape (for example, \u0042 ). ESC is the default format.
- **NCR** Numeric Character Representation (for example, &#22823 or &#177 ; )
- **PAREN** Unicode Parenthesis Escape (for example, <u0061>)
- **UTF8** UTF8 encoding.
- **UTF16** UTF16 encoding with big endian. UCS2 is an alias.
- **UTF16B** UTF16 encoding with big endian. UCS2B is an alias.
- **UTF16L** UTF16 encoding with little endian. UCS2L is an alias.
- **UTF32** UTF32 encoding with big endian. UCS4 is an alias.
- **UTF32B** UTF32 encoding with big endian. UCS4B is an alias.
- **UTF32L** UTF32 encoding with little endian. UCS4L is an alias.

**Details**

This function reads Unicode characters and converts them to the current SAS session encoding.

The Unicode function supports the Unicode supplementary characters.
Example

The following example demonstrates the functionality of the UNICODE function:

```sas
/* Run this program using any CJK LOCALE */
data _null_;  
str1=unicode("\u0041\u0042\u0043");put str1=;          /* default */
str2=unicode("\u0041\u0042\u0043", "esc");put str2=;     /* ESC - Unicode Escape */
str3=unicode("&177;", "ncr");put str3=;              /* NCR - Numeric Character Representation */
str4=unicode("&22823;", "ncr");put str4=;              /* NCR - Numeric Character Representation */
str5=unicode('<u0061><u0062>', 'paren');put str5=;   /* PAREN - Unicode Parenthesis Escape */
str6=unicode('2759'x, 'ucs2');put str6=;               /* UCS2 - UCS2 encoding */
str7=unicode('5927'x, 'ucs2b');put str7=;             /* UCS2B - UCS2 encoding with big endian */
str8=unicode('2759'x, 'ucs2l');put str8=;             /* UCS2L - UCS2 encoding with little endian */
str9=unicode('27590000'x,'ucs4');put str9=;            /* UCS4 - UCS4 encoding */
str10=unicode('00005927'x,'ucs4b');put str10=;         /* UCS4B - UCS4 encoding with big endian */
str11=unicode('27590000'x,'ucs4l');put str11=;         /* UCS4L - UCS4 encoding with little endian */
str12=unicode('E5AA7'x, 'utf8');put str12=;           /* UTF8 - UTF8 encoding */
str13=unicode('2759'x, 'utf16');put str13=;            /* UTF16 - UTF16 encoding */
str14=unicode('5927'x, 'utf16b');put str14=;           /* UTF16B - UTF16 encoding with big endian */
str15=unicode('2759'x, 'utf16l');put str15=;            /* UTF16L - UTF16 encoding with little endian */
str16=unicode("\u00020bb7", "esc");put str16=;        /* ESC - Unicode Escape for Supplementary Character */
str17=unicode("%134071;", "ncr");put str17=;           /* NCR - Numeric Character Representation for Supplementary Character */
str18=unicode('<u00020BB7>', 'paren');put str18=;      /* PAREN - Unicode Parenthesis Escape for Supplementary Character */
run;
```

Here are the results from the UNICODE function example:

| str1=ABC |
| str2=ABC |
| str3=|
| str4=大 |
| str5=ab |
| str6=大 |
| str7=大 |
| str8=大 |
| str9=大 |
| str10=大 |
| str11=大 |
| str12=大 |
| str13=大 |
| str14=大 |
| str15=大 |

UNICODEC Function

Converts characters in the current SAS session encoding to Unicode characters.

Category: Character

Restriction: This function is assigned an I18N Level 2 status and designed for use with SBCS, DBCS, and MBCS (UTF8). For more information, see Internationalization Compatibility on page 307.
Syntax

\[ \text{STR}=\text{UNICODEC}(\langle \text{instr}\rangle,\langle \text{Unicode type}\rangle) \]

**Required Arguments**

- **str**
  - data string that has been converted to Unicode encoding.

- **instr**
  - input data string.

- **Unicode type**
  - Unicode character formats
    - ESC: Unicode Escape (for example, \u0042 ) ESC is the default format.
    - NCR: Numeric Character Representation (for example, &\#22823 or &\#177 ;)
    - PAREN: Unicode Parenthesis Escape (for example, &lt;u0061&gt;)
    - UTF8: UTF8 encoding.
    - UTF16: UTF16 encoding with big endian. UCS2 is an alias.
    - UTF16B: UTF16 encoding with big endian. UCS2B is an alias.
    - UTF16L: UTF16 encoding with little endian. UCS2L is an alias.
    - UTF32: UTF32 encoding with big endian. UCS4 is an alias.
    - UTF32B: UTF32 encoding with big endian. UCS4B is an alias.
    - UTF32L: UTF32L encoding with big endian. UCS4L is an alias.

**Details**

This function reads characters that are in the current SAS session encoding and converts them to Unicode encoding.

**Example**

The following example demonstrates the functionality of the UNICODEC function:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>str1=unicodc(&quot;ABC&quot;, 'utf8');</td>
<td>str1=414243202020</td>
</tr>
<tr>
<td>put str1= $hex12.;</td>
<td></td>
</tr>
<tr>
<td>str2=unicodc(&quot;ABCé&quot;, 'utf8');</td>
<td>str2=414243C3A920</td>
</tr>
<tr>
<td>put str2= $hex12.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**UNICODELEN Function**

Specifies the length of the character unit for the Unicode data.
Syntax
UNICODELEN()

Details
The UNICODELEN function specifies the length of the character unit for the UNICODE data.

Example
This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1=unicodelen(&quot;abcä&quot;);</td>
<td>len1=4</td>
</tr>
<tr>
<td>len2=unicodelen(&quot;\0041\0042\0043\5927&quot;,&quot;esc&quot;);</td>
<td>len2=4</td>
</tr>
<tr>
<td>len3=unicodelen(&quot;&amp;22823;&quot;,&quot;ncr&quot;);</td>
<td>len3=1</td>
</tr>
<tr>
<td>len4=unicodelen(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;;'paren&quot;);</td>
<td>len4=2</td>
</tr>
</tbody>
</table>

See Also

Functions:
- “UNICODEWIDTH Function” on page 416

UNICODEWIDTH Function
Specifies the length of a display unit for the Unicode data.

Syntax
UNICODEWIDTH()
Details

The UNICODEWIDTH function specifies the length of a display unit for the Unicode data. The display unit displays the width of a character when the character is displayed with fixed width font.

The characters in CJK Symbols, CJK Unified Ideographs, Full width Alphabets and Punctuation, Full width currency symbols, CJK Unified Ideograph Extension B-D and CJK Compatibility Ideographs Supplement have the value of a display unit 2. Other characters are display unit 1.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1=unicodewidth(&quot;abc衆&quot;);</td>
<td>len1=5</td>
</tr>
<tr>
<td>len2=unicodewidth(&quot;\u0041\u0042\u0043\u5927&quot;,&quot;esc&quot;);</td>
<td>len2=5</td>
</tr>
<tr>
<td>len3=unicodewidth(&quot;大 &quot;,&quot;ncr&quot;);</td>
<td>len3=2</td>
</tr>
<tr>
<td>len4=unicodewidth(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;,&quot;paren&quot;);</td>
<td>len4=2</td>
</tr>
</tbody>
</table>

See Also

Functions:

• “UNICODELEN Function” on page 415

VARTRANSCODE Function

Returns the transcode attribute of a SAS data set variable.

Category: Variable Information

Syntax

VARTRANSCODE(data-set-id, var-num)

Required Arguments

data-set-id

specifies the data set identifier that the OPEN function returns.

var-num

specifies the position of the variable in the SAS data set.

Tip The VARNUM function returns this value.
Details

Transcoding is the process of converting data from one encoding to another. The VTRANSCODE function returns 0 if the \texttt{var-num} variable does not transcode its value, or 1 if the \texttt{var-num} variable transcodes its value.


Example

The following example shows how to determine whether a character variable is transcoded:

```sas
data a;
  attrib x length=$3. transcode=no;
  attrib y length=$3. transcode=yes;
  x='abc';
  y='xyz';
run;

data _null_;
  dsid=open('work.a','i');
  nobs=attrn(dsid,"nobs");
  nvars=attrn(dsid,"nvars");
  do i=1 to nobs;
    xrc=fetch(dsid,1);
    do j=1 to nvars;
      transcode = vartranscode(dsid,j);
      put transcode=;
    end;
  end;
run;
```

SAS writes the following output to the log:

```
transcode=0
transcode=1
```

See Also

Functions:

- “ATTRN Function” in SAS Functions and CALL Routines: Reference
- “OPEN Function” in SAS Functions and CALL Routines: Reference
- “VARNUM Function” in SAS Functions and CALL Routines: Reference
- “VTRANSCODE Function” on page 419
- “VTRANSCODEX Function” on page 420
VTRANSCODE Function
Returns a value that indicates whether transcoding is enabled for the specified character variable.

Category: Variable Information

Syntax
VTRANSCODE (var)

Required Argument
var
specifies a character variable that is expressed as a scalar or as an array reference.

Restriction
You cannot use an expression as an argument.

Details
The VTRANSCODE function returns 0 if transcoding is off, and 1 if transcoding is on. By default, all character variables in the DATA step are transoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

Comparisons
- The VTRANSCODE function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however, evaluates the argument to determine the variable name. The function then returns the transcoding status (on or off) that is associated with that variable name.
- The VTRANSCODE function does not accept an expression as an argument. The VTRANSCODEX function accepts expressions, but the value of the specified expression cannot denote an array reference.
- Related functions return the value of other variable attributes, such as the variable name, type, format, and length. For a list of the variable attributes, see the “Variable Information” functions in SAS Functions and CALL Routines: Reference.

Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrib x transcode = yes;</td>
<td></td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td></td>
</tr>
<tr>
<td>rcl = vtranscode[y];</td>
<td>rcl=0</td>
</tr>
<tr>
<td>put rcl=;</td>
<td></td>
</tr>
</tbody>
</table>
VTRANSCODEX Function

Returns a value that indicates whether transcoding is enabled for the specified argument.

**Category:** Variable Information

**Syntax**

VTRANSCODEX (var)

**Required Argument**

*var*

specifies any SAS character expression that evaluates to a character variable name.

**Restriction**

The value of the specified expression cannot denote an array reference.

**Details**

The VTRANSCODEX function returns 0 if transcoding is off, and 1 if transcoding is on. By default, all character variables in the DATA step are transcoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

**Comparisons**

- The VTRANSCODE function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however, evaluates the argument to determine the variable name. The function then returns the transcoding status (on or off) that is associated with that variable name.
- The VTRANSCODE function does not accept an expression as an argument. The VTRANSCODEX function accepts expressions, but the value of the specified expression cannot denote an array reference.
- Related functions return the value of other variable attributes, such as the variable name, type, format, and length. For a list of the variable attributes, see the “Variable Information” functions in *SAS Functions and CALL Routines: Reference*. 

**See Also**

**Functions:**

- “VTRANSCODEX Function” on page 420

**Statements:**

- ATTRIB in
Example

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrib x transcode = yes;</td>
<td></td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td></td>
</tr>
<tr>
<td>rc1 = vtranscodex('y');</td>
<td></td>
</tr>
<tr>
<td>put rc1=;</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>--------</td>
</tr>
</tbody>
</table>

rc1=0

See Also

Functions:

• “VTRANSCODE Function” on page 419

Statements:

• ATTRIB
Part 6

Informats for NLS

Chapter 13

Dictionary of Informats for NLS

…………………………………… 425
Chapter 13
Dictionary of Informats for NLS

Informat by Category

Dictionary

$CPTDWw. Informat
$CPTWDw. Informat
EUROw.d Informat
EUROXw.d Informat
$KANJw. Informat
$KANJIXw. Informat
$LOGVSw. Informat
$LOGVSRw. Informat
MINGUOw. Informat
NENGOw. Informat
NLDATExw. Informat
NLDATExWw. Informat
NLDATMw. Informat
NLDATMWw. Informat
NLMNIAEDw.d Informat
NLMNIAUDw.d Informat
NLMNIBGNw.d Informat
NLMNIBRLw.d Informat
NLMNICADw.d Informat
NLMNICFw.d Informat
NLMNICNYw.d Informat
NLMNICZKw.d Informat
NLMNIDKWw.d Informat
NLMNIEEKw.d Informat
NLMNIEGPw.d Informat
NLMNIEURw.d Informat
NLMNGBPw.d Informat
NLMNIIHKDw.d Informat
NLMNIIHRKw.d Informat
NLMNIIHUFw.d Informat
NLMNIIDRw.d Informat
NLMNIILSw.d Informat
NLMNIINRw.d Informat
NLMNIIKPw.d Informat
NLMNIIKLw.d Informat
NLMNIIKRWw.d Informat
NLMNILTLw.d Informat
NLMNILVLw.d Informat
NLMNIMOPw.d Informat
<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLMNIMXNw.d Informat</td>
<td></td>
<td>476</td>
</tr>
<tr>
<td>NLMNIMYRw.d Informat</td>
<td></td>
<td>477</td>
</tr>
<tr>
<td>NLMNINOKw.d Informat</td>
<td></td>
<td>478</td>
</tr>
<tr>
<td>NLMNINZDw.d Informat</td>
<td></td>
<td>479</td>
</tr>
<tr>
<td>NLMNIPLNw.d Informat</td>
<td></td>
<td>480</td>
</tr>
<tr>
<td>NLMNIRUBw.d Informat</td>
<td></td>
<td>481</td>
</tr>
<tr>
<td>NLMNISEKw.d Informat</td>
<td></td>
<td>482</td>
</tr>
<tr>
<td>NLMNISGDw.d Informat</td>
<td></td>
<td>483</td>
</tr>
<tr>
<td>NLMNITHBw.d Informat</td>
<td></td>
<td>484</td>
</tr>
<tr>
<td>NLMNITRYw.d Informat</td>
<td></td>
<td>485</td>
</tr>
<tr>
<td>NLMNITWDw.d Informat</td>
<td></td>
<td>486</td>
</tr>
<tr>
<td>NLMNIUSDw.d Informat</td>
<td></td>
<td>487</td>
</tr>
<tr>
<td>NLMNIZARw.d Informat</td>
<td></td>
<td>488</td>
</tr>
<tr>
<td>NLMNLAEDw.d Informat</td>
<td></td>
<td>489</td>
</tr>
<tr>
<td>NLMNLAUDw.d Informat</td>
<td></td>
<td>490</td>
</tr>
<tr>
<td>NLMNLBGNw.d Informat</td>
<td></td>
<td>491</td>
</tr>
<tr>
<td>NLMNLBRLw.d Informat</td>
<td></td>
<td>492</td>
</tr>
<tr>
<td>NLMNLCDw.d Informat</td>
<td></td>
<td>493</td>
</tr>
<tr>
<td>NLMNLCHFw.d Informat</td>
<td></td>
<td>494</td>
</tr>
<tr>
<td>NLMNLCNNyw.d Informat</td>
<td></td>
<td>495</td>
</tr>
<tr>
<td>NLMNLCKZw.d Informat</td>
<td></td>
<td>496</td>
</tr>
<tr>
<td>NLMNLDKKw.d Informat</td>
<td></td>
<td>497</td>
</tr>
<tr>
<td>NLMNLEEKw.d Informat</td>
<td></td>
<td>498</td>
</tr>
<tr>
<td>NLMNLEGpw.d Informat</td>
<td></td>
<td>499</td>
</tr>
<tr>
<td>NLMNLEURw.d Informat</td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>NLMNLGBPw.d Informat</td>
<td></td>
<td>501</td>
</tr>
<tr>
<td>NLMNLHKDw.d Informat</td>
<td></td>
<td>502</td>
</tr>
<tr>
<td>NLMNLHKRw.d Informat</td>
<td></td>
<td>503</td>
</tr>
<tr>
<td>NLMNLHUFw.d Informat</td>
<td></td>
<td>504</td>
</tr>
<tr>
<td>NLMNLIDRw.d Informat</td>
<td></td>
<td>505</td>
</tr>
<tr>
<td>NLMNLIISw.d Informat</td>
<td></td>
<td>506</td>
</tr>
<tr>
<td>NLMNLINRw.d Informat</td>
<td></td>
<td>507</td>
</tr>
<tr>
<td>NLMNLJPYw.d Informat</td>
<td></td>
<td>508</td>
</tr>
<tr>
<td>NLMNLKRw.d Informat</td>
<td></td>
<td>509</td>
</tr>
<tr>
<td>NLMNLTTv.d Informat</td>
<td></td>
<td>510</td>
</tr>
<tr>
<td>NLMNLTVw.d Informat</td>
<td></td>
<td>511</td>
</tr>
<tr>
<td>NLMNLMPw.d Informat</td>
<td></td>
<td>512</td>
</tr>
<tr>
<td>NLMNLMXw.d Informat</td>
<td></td>
<td>513</td>
</tr>
<tr>
<td>NLMNLMYRw.d Informat</td>
<td></td>
<td>514</td>
</tr>
<tr>
<td>NLMNLNOKw.d Informat</td>
<td></td>
<td>515</td>
</tr>
<tr>
<td>NLMNLNZDw.d Informat</td>
<td></td>
<td>516</td>
</tr>
<tr>
<td>NLMNLPLNw.d Informat</td>
<td></td>
<td>517</td>
</tr>
<tr>
<td>NLMNLRubw.d Informat</td>
<td></td>
<td>518</td>
</tr>
<tr>
<td>NLMNISEKw.d Informat</td>
<td></td>
<td>519</td>
</tr>
<tr>
<td>NLMNLSGDw.d Informat</td>
<td></td>
<td>520</td>
</tr>
<tr>
<td>NLMNITHDw.d Informat</td>
<td></td>
<td>521</td>
</tr>
<tr>
<td>NLMNLTRYw.d Informat</td>
<td></td>
<td>522</td>
</tr>
<tr>
<td>NLMNLWDb.d Informat</td>
<td></td>
<td>523</td>
</tr>
<tr>
<td>NLMNLUSDw.d Informat</td>
<td></td>
<td>524</td>
</tr>
<tr>
<td>NLMNIZARw.d Informat</td>
<td></td>
<td>525</td>
</tr>
<tr>
<td>NLMNYw.d Informat</td>
<td></td>
<td>526</td>
</tr>
<tr>
<td>NLMNYw.d Informat</td>
<td></td>
<td>527</td>
</tr>
<tr>
<td>NLMNUMw.d Informat</td>
<td></td>
<td>529</td>
</tr>
<tr>
<td>NLMNUMw.d Informat</td>
<td></td>
<td>530</td>
</tr>
<tr>
<td>NLPCTw.d Informat</td>
<td></td>
<td>531</td>
</tr>
<tr>
<td>NLPCTw.d Informat</td>
<td></td>
<td>532</td>
</tr>
<tr>
<td>NLPCTw.d Informat</td>
<td></td>
<td>533</td>
</tr>
</tbody>
</table>
There are six categories of SAS informats that support NLS:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI Text Handling</td>
<td>Instructs SAS to read bidirectional data values from data variables.</td>
</tr>
<tr>
<td>Character</td>
<td>Instructs SAS to read character data values into character variables.</td>
</tr>
<tr>
<td>DBCS</td>
<td>Instructs SAS to manage various Asian languages.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>Instructs SAS to read data values into variables that represent dates, times, and datetimes.</td>
</tr>
<tr>
<td>Hebrew Text Handling</td>
<td>Instructs SAS to read Hebrew data from data variables.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Instructs SAS to read numeric data values into numeric variables.</td>
</tr>
</tbody>
</table>
The following table provides brief descriptions of the SAS informats. For more detailed descriptions, see the NLS entry for each informat.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI Text Handling</td>
<td>$LOGVS_w$. Informat (p. 442)</td>
<td>Reads a character string that is in left-to-right logical order, and then converts the character string to visual order.</td>
</tr>
<tr>
<td></td>
<td>$LOGVSR_w$. Informat (p. 443)</td>
<td>Reads a character string that is in right-to-left logical order, and then converts the character string to visual order.</td>
</tr>
<tr>
<td></td>
<td>$VSLOG_w$. Informat (p. 558)</td>
<td>Reads a character string that is in visual order, and then converts the character string to left-to-right logical order.</td>
</tr>
<tr>
<td></td>
<td>$VSLOGR_w$. Informat (p. 559)</td>
<td>Reads a character string that is in visual order, and then converts the character string to right-to-left logical order.</td>
</tr>
<tr>
<td>Character</td>
<td>$REVERJ_w$. Informat (p. 536)</td>
<td>Reads character data from right to left and preserves blanks.</td>
</tr>
<tr>
<td></td>
<td>$REVERSE_w$. Informat (p. 537)</td>
<td>Reads character data from right to left, and then left aligns the text.</td>
</tr>
<tr>
<td></td>
<td>$UCS2B_w$. Informat (p. 538)</td>
<td>Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UCS2BE_w$. Informat (p. 539)</td>
<td>Reads a character string that is in the encoding of the current SAS session and then converts the character string to big-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS2L_w$. Informat (p. 540)</td>
<td>Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UCS2LE_w$. Informat (p. 541)</td>
<td>Reads a character string that is in the encoding of the current SAS session and then converts the character string to little-endian, 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS2X_w$. Informat (p. 542)</td>
<td>Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UCS2XE_w$. Informat (p. 543)</td>
<td>Reads a character string that is in the encoding of the current SAS session and then converts the character string to 16-bit, UCS2, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UCS4B_w$. Informat (p. 544)</td>
<td>Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UCS4L_w$. Informat (p. 545)</td>
<td>Reads a character string that is encoded in little-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>$UCS4Xw. Informat (p. 546)</td>
<td>Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UCS4XEw. Informat (p. 548)</td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to 32-bit, UCS4, Unicode encoding.</td>
</tr>
<tr>
<td></td>
<td>$UESCw. Informat (p. 549)</td>
<td>Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UESCEw. Informat (p. 550)</td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UESC representation.</td>
</tr>
<tr>
<td></td>
<td>$UNCRw. Informat (p. 551)</td>
<td>Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UNCREw. Informat (p. 552)</td>
<td>Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.</td>
</tr>
<tr>
<td></td>
<td>$UPARENw. Informat (p. 553)</td>
<td>Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>$UPARENEw. Informat (p. 554)</td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UPAREN representation.</td>
</tr>
<tr>
<td></td>
<td>$UTF8Xw. Informat (p. 557)</td>
<td>Reads a character string that is encoded in UTF-8, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>MINGUOw. Informat (p. 444)</td>
<td>Reads dates in Taiwanese format.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>NENGOw. Informat (p. 446)</td>
<td>Reads Japanese date values in the form eyymmmd.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>NLDATEw. Informat (p. 447)</td>
<td>Reads the date value in the specified locale, and then converts the date value to the local SAS date value.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>NLDATEWw. Informat (p. 448)</td>
<td>Reads the date value in the specified locale and then converts the date value to the local SAS date and the day of the week.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>NLDATMw. Informat (p. 449)</td>
<td>Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>NLDATMAP Informat (p. 450)</td>
<td>Reads the date value in the specified locale, and then converts the date value to the local SAS datetime with either a.m. or p.m.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLDATMWw. Informat (p. 451)</td>
<td>Reads the date value in the specified locale and then converts the date value to the local SAS day of the week and the datetime.</td>
</tr>
<tr>
<td></td>
<td>NLTIMAPw. Informat (p. 534)</td>
<td>Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
<tr>
<td></td>
<td>NLTIMEw. Informat (p. 535)</td>
<td>Reads the time value in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
<tr>
<td>DBCS</td>
<td>$KANJIw. Informat (p. 440)</td>
<td>Removes shift code data from DBCS data.</td>
</tr>
<tr>
<td></td>
<td>$KANJIXw. Informat (p. 441)</td>
<td>Adds shift-code data to DBCS data.</td>
</tr>
<tr>
<td>Hebrew Text Handling</td>
<td>SCPTDWw. Informat (p. 435)</td>
<td>Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding.</td>
</tr>
<tr>
<td></td>
<td>SCPTWDw. Informat (p. 436)</td>
<td>Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding.</td>
</tr>
<tr>
<td>Numeric</td>
<td>EUROw.d Informat (p. 437)</td>
<td>Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.</td>
</tr>
<tr>
<td></td>
<td>EUROXw.d Informat (p. 439)</td>
<td>Reads numeric values and removes embedded characters in European currency.</td>
</tr>
<tr>
<td></td>
<td>NLMNIAEDw.d Informat (p. 452)</td>
<td>Reads the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>NLMNIAUDw.d Informat (p. 453)</td>
<td>Reads the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>NLMNIBGNw.d Informat (p. 454)</td>
<td>Reads the monetary format of the international expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>NLMNIBRLw.d Informat (p. 455)</td>
<td>Reads the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>NLMNICADw.d Informat (p. 456)</td>
<td>Reads the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>NLMNICHFw.d Informat (p. 457)</td>
<td>Reads the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>NLMNICNYw.d Informat (p. 458)</td>
<td>Reads the monetary format of the international expression for China.</td>
</tr>
<tr>
<td></td>
<td>NLMNICZKw.d Informat (p. 459)</td>
<td>Reads the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>NLMNIDKKw.d Informat (p. 460)</td>
<td>Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIEEK</strong> <em>w.d Informat (p. 461)</em>*</td>
<td>Reads the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIEGP</strong> <em>w.d Informat (p. 462)</em>*</td>
<td>Reads the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIEUR</strong> <em>w.d Informat (p. 463)</em>*</td>
<td>Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIGBP</strong> <em>w.d Informat (p. 464)</em>*</td>
<td>Reads the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIHKD</strong> <em>w.d Informat (p. 465)</em>*</td>
<td>Reads the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIHRK</strong> <em>w.d Informat (p. 466)</em>*</td>
<td>Reads the monetary format of the international expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIHUF</strong> <em>w.d Informat (p. 467)</em>*</td>
<td>Reads the monetary format of the international expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIIDR</strong> <em>w.d Informat (p. 468)</em>*</td>
<td>Reads the monetary format of the international expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIILS</strong> <em>w.d Informat (p. 469)</em>*</td>
<td>Reads the monetary format of the international expression for Israel.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIINR</strong> <em>w.d Informat (p. 470)</em>*</td>
<td>Reads the monetary format of the international expression for India.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIIJPY</strong> <em>w.d Informat (p. 471)</em>*</td>
<td>Reads the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIKRW</strong> <em>w.d Informat (p. 472)</em>*</td>
<td>Reads the monetary format of the international expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNILTL</strong> <em>w.d Informat (p. 473)</em>*</td>
<td>Reads the monetary format of the international expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNILVL</strong> <em>w.d Informat (p. 474)</em>*</td>
<td>Reads the monetary format of the international expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIMOP</strong> <em>w.d Informat (p. 475)</em>*</td>
<td>Reads the monetary format of the international expression for Macau.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIMXN</strong> <em>w.d Informat (p. 476)</em>*</td>
<td>Reads the monetary format of the international expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td><strong>NLMNIMYR</strong> <em>w.d Informat (p. 477)</em>*</td>
<td>Reads the monetary format of the international expression for Malaysia.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NLMNINOKw.d</td>
<td>Informat (p. 478)</td>
<td>Reads the monetary format of the international expression for Norway.</td>
</tr>
<tr>
<td>NLMNINNZDw.d</td>
<td>Informat (p. 479)</td>
<td>Reads the monetary format of the international expression for New Zealand.</td>
</tr>
<tr>
<td>NLMNIPLNw.d</td>
<td>Informat (p. 480)</td>
<td>Reads the monetary format of the international expression for Poland.</td>
</tr>
<tr>
<td>NLMNIRUBw.d</td>
<td>Informat (p. 481)</td>
<td>Reads the monetary format of the international expression for Russia.</td>
</tr>
<tr>
<td>NLMNISEKw.d</td>
<td>Informat (p. 482)</td>
<td>Reads the monetary format of the international expression for Sweden.</td>
</tr>
<tr>
<td>NLMNISGDw.d</td>
<td>Informat (p. 483)</td>
<td>Reads the monetary format of the international expression for Singapore.</td>
</tr>
<tr>
<td>NLMNITHBw.d</td>
<td>Informat (p. 484)</td>
<td>Reads the monetary format of the international expression for Thailand.</td>
</tr>
<tr>
<td>NLMNITRYw.d</td>
<td>Informat (p. 485)</td>
<td>Reads the monetary format of the international expression for Turkey.</td>
</tr>
<tr>
<td>NLMNITWDw.d</td>
<td>Informat (p. 486)</td>
<td>Reads the monetary format of the international expression for Taiwan.</td>
</tr>
<tr>
<td>NLMNIUSDw.d</td>
<td>Informat (p. 487)</td>
<td>Reads the monetary format of the international expression for Puerto Rico.</td>
</tr>
<tr>
<td>NLMNIZARw.d</td>
<td>Informat (p. 488)</td>
<td>Reads the monetary format of the international expression for South Africa.</td>
</tr>
<tr>
<td>NLMNLAEDw.d</td>
<td>Informat (p. 489)</td>
<td>Reads the monetary format of the local expression for the United Arab Emirates.</td>
</tr>
<tr>
<td>NLMNLAUDw.d</td>
<td>Informat (p. 490)</td>
<td>Reads the monetary format of the local expression for Australia.</td>
</tr>
<tr>
<td>NLMNLBGNgw.d</td>
<td>Informat (p. 491)</td>
<td>Reads the monetary format of the local expression for Bulgaria.</td>
</tr>
<tr>
<td>NLMNLBRLw.d</td>
<td>Informat (p. 492)</td>
<td>Reads the monetary format of the local expression for Brazil.</td>
</tr>
<tr>
<td>NLMNLCADw.d</td>
<td>Informat (p. 493)</td>
<td>Reads the monetary format of the local expression for Canada.</td>
</tr>
<tr>
<td>NLMNILCHFw.d</td>
<td>Informat (p. 494)</td>
<td>Reads the monetary format of the local expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td>NLMNLSCNYw.d</td>
<td>Informat (p. 495)</td>
<td>Reads the monetary format of the local expression for China.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLMNLCZK</td>
<td>w.d Informat (p. 496)</td>
<td>Reads the monetary format of the local expression for the Czech Republic.</td>
</tr>
<tr>
<td>NLMNLDKK</td>
<td>w.d Informat (p. 497)</td>
<td>Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.</td>
</tr>
<tr>
<td>NLMNLEEK</td>
<td>w.d Informat (p. 498)</td>
<td>Reads the monetary format of the local expression for Estonia.</td>
</tr>
<tr>
<td>NLMNLEGp</td>
<td>w.d Informat (p. 499)</td>
<td>Reads the monetary format of the local expression for Egypt.</td>
</tr>
<tr>
<td>NLMNLEUR</td>
<td>w.d Informat (p. 500)</td>
<td>Reads the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.</td>
</tr>
<tr>
<td>NLMNLGBP</td>
<td>w.d Informat (p. 501)</td>
<td>Reads the monetary format of the local expression for the United Kingdom.</td>
</tr>
<tr>
<td>NLMNLHKD</td>
<td>w.d Informat (p. 502)</td>
<td>Reads the monetary format of the local expression for Hong Kong.</td>
</tr>
<tr>
<td>NLMNLHRK</td>
<td>w.d Informat (p. 503)</td>
<td>Reads the monetary format of the local expression for Croatia.</td>
</tr>
<tr>
<td>NLMNLHUF</td>
<td>w.d Informat (p. 504)</td>
<td>Reads the monetary format of the local expression for Hungary.</td>
</tr>
<tr>
<td>NLMNLIDR</td>
<td>w.d Informat (p. 505)</td>
<td>Reads the monetary format of the local expression for Indonesia.</td>
</tr>
<tr>
<td>NLMNLI LS</td>
<td>w.d Informat (p. 506)</td>
<td>Reads the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td>NLMNLINR</td>
<td>w.d Informat (p. 507)</td>
<td>Reads the monetary format of the local expression for India.</td>
</tr>
<tr>
<td>NLMNLJPY</td>
<td>w.d Informat (p. 508)</td>
<td>Reads the monetary format of the local expression for Japan.</td>
</tr>
<tr>
<td>NLMNLKRW</td>
<td>w.d Informat (p. 509)</td>
<td>Reads the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td>NLMNLLTL</td>
<td>w.d Informat (p. 510)</td>
<td>Reads the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td>NLMNLLVL</td>
<td>w.d Informat (p. 511)</td>
<td>Reads the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td>NLMNLMOP</td>
<td>w.d Informat (p. 512)</td>
<td>Reads the monetary format of the local expression for Macau.</td>
</tr>
<tr>
<td>Language Elements</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>NLMNLMXNw.d Informat (p. 513)</td>
<td>Reads the monetary format of the local expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>NLMNLMYRw.d Informat (p. 514)</td>
<td>Reads the monetary format of the local expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNOKw.d Informat (p. 515)</td>
<td>Reads the monetary format of the local expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>NLMNLNZDw.d Informat (p. 516)</td>
<td>Reads the monetary format of the local expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>NLMNLPawl.d Informat (p. 517)</td>
<td>Reads the monetary format of the local expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>NLMNLRUBw.d Informat (p. 518)</td>
<td>Reads the monetary format of the local expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>NLMNLSEKw.d Informat (p. 519)</td>
<td>Reads the monetary format of the local expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>NLMNLSGDw.d Informat (p. 520)</td>
<td>Reads the monetary format of the local expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTHBw.d Informat (p. 521)</td>
<td>Reads the monetary format of the local expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTRYw.d Informat (p. 522)</td>
<td>Reads the monetary format of the local expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>NLMNLTWDw.d Informat (p. 523)</td>
<td>Reads the monetary format of the local expression for Taiwan.</td>
<td></td>
</tr>
<tr>
<td>NLMNLUSDw.d Informat (p. 524)</td>
<td>Reads the monetary format of the local expression for Puerto Rico, and the United States.</td>
<td></td>
</tr>
<tr>
<td>NLMNLZARw.d Informat (p. 525)</td>
<td>Reads the monetary format of the local expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>NLMNYw.d Informat (p. 526)</td>
<td>Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLMNYIw.d Informat (p. 527)</td>
<td>Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLMNUMw.d Informat (p. 529)</td>
<td>Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLMNUMIw.d Informat (p. 530)</td>
<td>Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>NLPCTw.d Informat (p. 531)</td>
<td>Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>NLPCTIw.d Informat (p. 533)</td>
<td>Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.</td>
</tr>
<tr>
<td></td>
<td>YENw.d Informat (p. 560)</td>
<td>Removes embedded yen signs, commas, and decimal points.</td>
</tr>
</tbody>
</table>

**Dictionary**

### $\text{CPTD}Ww. \text{ Informat}$

Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding.

- **Category:** Hebrew Text Handling

### Syntax

$\text{CPTD}Ww.$

### Syntax Description

$w$

specifies the width of the input field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1–32000</td>
</tr>
</tbody>
</table>

### Comparisons

The $\text{CPTD}Ww.$ informat performs processing that is opposite of the $\text{CPTWD}Ww.$ informat.

### Example

The following example uses the input value of 808182.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('808182',$cptdw6.);</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td>12N</td>
</tr>
</tbody>
</table>
$CPTWDw. Informat

Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding.

Category: Hebrew Text Handling

Syntax

$CPTWDw.

Syntax Description

\( \text{w} \)

specifies the width of the input field.

Default 200

Range 1–32000

Comparisons

The $CPTWDw. informat performs processing that is opposite of the $CPTDw. informat.

Example

The following example uses the input value of אבר.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>436</td>
</tr>
</tbody>
</table>

\[x=\text{input ('אבר',}$\text{cptwd6.});\]
\[\text{put } x;\]
See Also

Formats:
- “$CPTWDw. Format” on page 108
- “$CPTDWw. Format” on page 107

Informat:
- “$CPTDWw. Informat” on page 435

EUROw.d Informat

Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.

Category: Numeric

Syntax

EUROw.d

Syntax Description

\( w \)

specifies the width of the input field.

Default 6
Range 1–32

\( d \)

specifies the power of 10 by which to divide the value. If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Details

The EURO\( w.d \) informat reads numeric values and removes embedded euro symbols (E), commas, blanks, percent signs, hyphens, and close parentheses from the input data. A decimal point is assumed to be a separator between the whole number and the decimal portion. The EURO\( w.d \) informat converts an open parenthesis at the beginning of a field to a minus sign.

Comparisons

- The EURO\( w.d \) informat is similar to the EUROX\( w.d \) informat, but EUROX\( w.d \) reverses the roles of the decimal point and the comma. This convention is common in European countries.
• If no commas or periods appear in the input, then the EUROw.d and the EUROXw.d informats are interchangeable.

Example

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

```sas
data _null_;  
  input x euro10.;  
  put x=;  
  datalines;  
E1  
E1.23  
1.23  
1,234.56  
;  
run;  
SAS Log:  
x=1  
x=1.23  
x=1.23  
x=1234.56
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>input x euro10.; put x;</td>
<td>1</td>
</tr>
<tr>
<td>E1.23</td>
<td>input x euro10.; put x;</td>
<td>1.23</td>
</tr>
<tr>
<td>1.23</td>
<td>input x euro10.; put x;</td>
<td>1.23</td>
</tr>
<tr>
<td>1,234.56</td>
<td>input x euro10.; put x;</td>
<td>1234.56</td>
</tr>
</tbody>
</table>

See Also

Formats:
• “EUROw.d Format” on page 114
• “EUROXw.d Format” on page 117

Informat:
• “EUROXw.d Informat” on page 439
**EUROXw.d Informat**

Reads numeric values and removes embedded characters in European currency.

**Syntax**

\[ \text{EUROXw.d} \]

**Syntax Description**

\[ w \]

specifies the width of the input field.

- Default: 6
- Range: 1–32

\[ d \]

specifies the power of 10 by which to divide the value. If the data contains a comma, which represents a decimal point, the \( d \) value is ignored.

- Default: 0
- Range: 0–31

**Details**

The \( \text{EUROXw.d} \) informat reads numeric values and removes embedded euro symbols (E), periods, blanks, percent signs, hyphens, and close parentheses from the input data. A comma is assumed to be a separator between the whole number and the decimal portion. The \( \text{EUROXw.d} \) informat converts an open parenthesis at the beginning of a field to a minus sign.

**Comparisons**

- The \( \text{EUROXw.d} \) informat is similar to the \( \text{EUROw.d} \) informat, but \( \text{EUROw.d} \) reverses the roles of the comma and the decimal point. This convention is common in English-speaking countries.

- If no commas or periods appear in the input, the \( \text{EUROXw.d} \) and the \( \text{EUROw.d} \) informats are interchangeable.

**Example**

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

```
data _null_;  
  input x eurox10.;  
  put x=;  
  datalines;  
E1```
E1.23
1.23
1,234.56
; run;
SAS Log:
7  input x eurox10.;
8  put x=;
9  datalines;
x=1
x=123
x=123
x=1.23456

<table>
<thead>
<tr>
<th>Values</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>input x eurox10.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>E1.23</td>
<td>input x eurox10.;</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1.23</td>
<td>input x eurox10.;</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1,234.56</td>
<td>input x eurox10.;</td>
<td>1.23456</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “EUROw.d Format” on page 114
- “EUROXw.d Format” on page 117

Informat:
- “EUROw.d Informat” on page 437

$KANJIfw. Informat

Removes shift code data from DBCS data.

Category: DBCS

Syntax

$KANJIfw
Syntax Description

$w$

specifies the width of the input field.

Range

The minimum width for the informat is 2.

Restriction

The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.

Details

The $\text{KANJI}$ informat removes shift-code data from DBCS data. The $\text{KANJI}$ informat processes host-mainframe data. $\text{KANJI}$ can be used on other platforms. If you use the $\text{KANJI}$ informat on non-EBCDIC (non-modal encoding) hosts, the data does not change.

The data must start with SO and end with SI, unless single-byte blank data are returned. The input data length must be $2 + \text{(SO/SI length)} \times 2$.

See Also

Formats:

- “$\text{KANJI}w. Format$” on page 121
- “$\text{KANJI}xw. Format$” on page 122

Informat:

- “$\text{KANJI}xw. Informat$” on page 441

$\text{KANJI}xw. Informat$

Adds shift-code data to DBCS data.

Category: DBCS

Syntax

$\text{KANJI}xw$

Syntax Description

$w$

specifies the width of the input field.

Range

The minimum width for the informat is $2 + \text{(length of shift code used on the current DBCSTYPE= setting)} \times 2$.

Restriction

The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.
Details

The $KANJIX informat adds shift-code data to DBCS data that does not have shift-code data. If the input data is blank, shift-code data is not added. The $KANJIX informat processes host-mainframe data, but $KANJIX can be used on other platforms. If you use the $KANJIX informat on non-EBCDIC (non-modal encoding) hosts, the data does not change.

See Also

Formats:

• “$KANJIw. Format” on page 121
• “$KANJIXw. Format” on page 122

Informat:

• “$KANJIw. Informat” on page 440

$LOGVSw. Informat

Reads a character string that is in left-to-right logical order, and then converts the character string to visual order.

Category: BIDI Text Handling

Syntax

$LOGVSw.

Syntax Description

w

• specifies the width of the input field.

Default 200

Range 1–32000

Comparisons

The $LOGVSw. informat performs processing that is opposite to the LOGVSRw. informat.

Example

The following example uses the Hebrew input value of “_flight.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- - - - - - 1 - - - -</td>
</tr>
</tbody>
</table>
The following example uses the Arabic input value of "الحاسوب.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('الحاسوب',$logvs12.); put x;</td>
<td>الحاسوب</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$LOGVSRW. Format” on page 124
- “$LOGVSW. Format” on page 123

Informat:
- “$LOGVSRW. Informat” on page 443

$LOGVSRW. Informat

Reads a character string that is in right-to-left logical order, and then converts the character string to visual order.

Category: BIDI Text Handling

Syntax

$LOGVSRW.

Syntax Description

w

specifies the width of the input field.

Default 200

Range 1–32000
Comparisons

The $LOGVSR_w$ informat performs processing that is opposite to the $LOGVSw_w$ informat.

Example

The following example uses the Hebrew input value of "_flight."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('_flight',Slogsr12.); put x;</td>
<td>flight נסיעה</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of "_computer."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('_computer',Slogsr12.); put x;</td>
<td>computer</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$LOGVSw. Format” on page 123
- “$LOGVSRw. Format” on page 124

Informat:
- “$LOGVSw. Informat” on page 442

MINGUOW. Informat

Reads dates in Taiwanese format.

Category: Date and Time

Syntax

MINGUOW.
Syntax Description

w
   specifies the width of the input field.
   Default  6
   Range    6–10

Details

The general form of a Taiwanese date is yyyyymmdd:

  yyyy
    is an integer that represents the year.

  mm
    is an integer from 01 through 12 that represents the month.

  dd
    is an integer from 01 through 31 that represents the day of the month.

The Taiwanese calendar uses 1912 as the base year (01/01/01 is January 1, 1912). Dates before 1912 are not valid. Year values do not roll over after 100 years. Instead, they continue to increase.

You can separate the year, month, and day values with any delimiters, such as blanks, slashes, or hyphens, that are permitted by the YYMMDDw. informat. If delimiters are used, place them between all the values. If you omit delimiters, be sure to use a leading zero for days or months that have a value less than 10.

Example

The following examples use different dates for input values.

```plaintext
input date minguo10.;
put date date9.;
data _null_;  
   input date minguo10.;
   put date date9.;
datalines;
49/01/01
891215
03-01-01
;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>49/01/01</td>
<td>01JAN1960</td>
</tr>
<tr>
<td>891215</td>
<td>15DEC2000</td>
</tr>
<tr>
<td>103-01-01</td>
<td>01JAN2014</td>
</tr>
</tbody>
</table>
See Also

Format:
- “MINGUOw. Format” on page 126

Informat:
- “YYMMDDw. Informat” in SAS Formats and Informats: Reference

**NENGOw. Informat**

Reads Japanese date values in the form *eyymmdd*.

**Category:** Date and Time

**Syntax**

**NENGOw:**

**Syntax Description**

*w*

specifies the width of the input field.

**Default** 10

**Range** 7–32

**Details**

The general form of a Japanese date is *eyymmdd*:

*e*

is the first letter of the name of the imperial era (Meiji, Taisho, Showa, or Heisei).

*yy*

is an integer that represents the year.

*mm*

is an integer from 01 through 12 that represents the month.

*dd*

is an integer from 01 through 31 that represents the day of the month.

The *e* value can be separated from the integers by a period. If you omit *e*, SAS uses the current imperial era. You can separate the year, month, and day values by blanks or any nonnumeric character. However, if delimiters are used, place them between all the values. If you omit delimiters, be sure to use a leading zero for days or months that are values less than 10.

**Example**

The following examples use different input values.

```sas
input nengo_date nengo8.;
```
put nengo_date date9.;
data _null_;  
   input nengo_date nengo8.;  
   put nengo_date date9.;  
   put nengo_date= ;  
   datalines;
h11108  
h.11108  
11/10/08  
;

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>h11108</td>
<td>08OCT1999</td>
</tr>
<tr>
<td>h.11108</td>
<td>08OCT1999</td>
</tr>
<tr>
<td>11/10/08</td>
<td>08OCT1999</td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**
- “NENGOw. Format” on page 127

---

**NLDATEw. Informat**

Reads the date value in the specified locale, and then converts the date value to the local SAS date value.

**Category:** Date and Time  
**Alias:** NLDATEW

**Syntax**

\texttt{NLDATEw.}

**Syntax Description**

\texttt{w}

specifies the width of the input field.

Default \hspace{1cm} 20  
Range \hspace{1cm} 10–200
### Example

The following examples use the input February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----+-----+1-----+</td>
<td></td>
</tr>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>15760</td>
</tr>
<tr>
<td>dy='February 24, 2003';</td>
<td></td>
</tr>
<tr>
<td>y=input('dy,nldate200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>-----+-----+1-----+</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>15760</td>
</tr>
<tr>
<td>dy='24. Februar 2003';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldate16.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLDATEw. Format” on page 130

### NLDATEWw. Informat

Reads the date value in the specified locale and then converts the date value to the local SAS date and the day of the week.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

### Syntax

**NLDATEWw.**

### Syntax Description

**w**

- specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>16–200</td>
</tr>
</tbody>
</table>
Example

The following examples use the input February 24, 2014.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>19778</td>
</tr>
<tr>
<td>data;</td>
<td></td>
</tr>
<tr>
<td>dy='Monday, February 24, 2014';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldatew200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>19778</td>
</tr>
<tr>
<td>dy='Mo. 24.Februar 2014';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldatew16.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMw. Informat

Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value.

**Category:** Date and Time

**Alias:** NLDATMW, NLDATMAP

Syntax

NLDATMw.

**Syntax Description**

w

specifies the width of the input field.

- **Default:** 19
- **Range:** 19–200

Example

The following examples use the input value of February 24, 2003 12:39:43.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
Statements | Results
---|---
oneoptions locale=English_UnitedStates;\n\ny=input(’24.Feb03:12:39:43’, nldatm.);\nput y=;\n
go\n
1361709583\n
Oneoptions locale=German_Germany;\n\ny=input(’24.Februar 2003 12.39 Uhr’, nldatm.);\nput y=;\n
1330171200

See Also

Format:

• “NLDATMw. Format” on page 151

NLDATMAP Informat

Reads the date value in the specified locale, and then converts the date value to the local SAS datetime with either a.m. or p.m.

Category: Date and Time
Alignment: Left

Syntax

NLDATMAPW;

Syntax Description

W

specifies the width of the output field. If necessary, SAS abbreviates the datetime value to fit the format width.

Default 32
Range $17–200$

Example

These examples use the input value of 12:39:43 p.m. on February 24, 2014.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---+---</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

---
## NLDATMWw. Informat

Reads the date value in the specified locale and then converts the date value to the local SAS day of the week and the datetime.

**Category:** Date and Time  
**Alignment:** Left

### Syntax

```
NLDATMWw
```

### Syntax Description

`w`

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>34–200</td>
</tr>
</tbody>
</table>

### Example

The following examples use the input Mon, Feb 24, 2014 12:39:43 PM.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>data;</td>
<td></td>
</tr>
<tr>
<td>dy='February 24, 2014 12:39:43 PM';</td>
<td>1708864783</td>
</tr>
<tr>
<td>y=input(dy,nldatmap200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td>options locale=Spanish_Mexico;</td>
<td>1708864783</td>
</tr>
<tr>
<td>data;</td>
<td></td>
</tr>
<tr>
<td>dy='24/02/2003 12:39:43 PM';</td>
<td></td>
</tr>
<tr>
<td>y=input(dy,nldatmap200.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
NLMNIAED\(w.d\) Informat

Reads the monetary format of the international expression for the United Arab Emirates.

**Category:** Numeric  
**Alignment:** Left

**Syntax**

\[
\text{NLMNIAED}\(w.d\)
\]

**Syntax Description**

\(w\)

- specifies the width of the output field.
  - Default: 9
  - Range: 1–32

\(d\)

- specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 0
  - Range: 0–31

**Example**

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

\[
\begin{align*}
\text{x} &= \text{input}'\{($12,345.67)'\}, \text{nlmniaed32.2}; \\
\text{y} &= \text{input}'\{($12,345.67)'\}, \text{dollar32.2};
\end{align*}
\]
See Also

Informat:
- “NLMNLAEDw.d Informat” on page 489

NLMNIAUDw.d Informat
Reads the monetary format of the international expression for Australia.

Category: Numeric
Alignment: Left

Syntax
NLMNIAUDw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{input}(\' \langle \$12,345.67 \rangle \', \text{nlmniaud32.2});
y = \text{input}(\' \langle \$12,345.67 \rangle \', \text{dollar32.2});
\]
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:
- “NLMNLAUDw.d Informat” on page 490

**NLMNIBGNw.d Informat**

Reads the monetary format of the international expression for Bulgaria.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNIBGNw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default:** 0
- **Range:** 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put('(-1234.56789)',nlmnibgn32.2);
y=put('(-1234.56789)',dollar32.2);
```
See Also

Informat:

- “NLMNLBGNw.d Informat” on page 491

### NLMNIBRLw.d Informat

Reads the monetary format of the international expression for Brazil.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

\[\text{NLMNIBRLw.d}\]

#### Syntax Description

- \(w\) specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- \(d\) specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - **Default:** 0
  - **Range:** 0–31

#### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

\[\begin{align*}
x &= \text{input}('($12,345.67)',\text{nlmnibr132.2}); \\
y &= \text{input}('($12,345.67)',\text{dollar32.2});
\end{align*}\]
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

Informat:
- “NLMNLBRLw.d Informat” on page 492

---

### NLMNICADw.d Informat

Reads the monetary format of the international expression for Canada.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[ \text{NLMNICAD}w.d \]

### Syntax Description

- \( w \) specifies the width of the output field.
  - Default 9  
  - Range 1–32

- \( d \) specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.
  - Default 0  
  - Range 0–31

### Example

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnicad32.2);
y=input'($12,345.67)',dollar32.2);
```
### NLMNICHFw.d Informat

Reads the monetary format of the international expression for Liechtenstein and Switzerland.

**Category:** Numeric  
**Alignment:** Left

#### Syntax

\[ \text{NLMNICHF} \text{w.d} \]

#### Syntax Description

- **w**  
  Specifies the width of the output field.  
  - Default: 9  
  - Range: 1–32

- **d**  
  Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.  
  - Default: 0  
  - Range: 0–31

#### Example

In the following example, the `LOCALE=` system option is set to English_UnitedStates.

\[
\begin{align*}
  x &= \text{input}'($12,345.67)', \text{nlmnichf32.2}); \\
  y &= \text{input}'($12,345.67)'\text{dollar32.2});
\end{align*}
\]
### NLMNICNYw.d Informat

Reads the monetary format of the international expression for China.

**Category:** Numeric  
**Alignment:** Left

**Syntax**

\[ \text{NLMNICNY}w.d \]

**Syntax Description**

- \( w \)  
  - Specifies the width of the output field.  
  - Default: 9  
  - Range: 1–32

- \( d \)  
  - Specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.  
  - Default: 0  
  - Range: 0–31

**Example**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input'($12,345.67)',nlmnicny32.2);
y=input'($12,345.67)\text{dollar32.2}');
```

---

### See Also

- “NLMNICHFw.d Format” on page 181
NLMNICZKw.d Informat

Reads the monetary format of the international expression for the Czech Republic.

**Category:** Numeric  
**Alignment:** Left

**Syntax**

NLMNICZKw.d

**Syntax Description**

- $w$
  - specifies the width of the output field.
  - Default: 9  
  - Range: 1–32

- $d$
  - specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 0  
  - Range: 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmniczk32.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

Informat:
- [“NLMNLCZKw.d Informat” on page 496](#)

---

### NLMNIDKKw.d Informat

Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

**NLMNIDKKw.d**

#### Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 9
  - Range: 1–32

- **d**
  - specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 0
  - Range: 0–31

#### Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```plaintext
x=input('($12,345.67)',nlmndkk32.2);
y=input'($12,345.67)',dollar32.2);
```
Statements | Results
-----------|--------

| put x=;   | -12345.67 |
| put y=;   | -12345.67 |

See Also

Format:
- “NLMNIDKKw.d Format” on page 184

NLMNIEEKw.d Informat
Reads the monetary format of the international expression for Estonia.

Category: Numeric
Alignment: Left

Syntax
NLMNIEEKw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmniek32.2);
y=input('($12,345.67)',dollar32.2);
Statements | Results
---|---
| ---+----1----+
| put x=; | -12345.67
| put y=; | -12345.67

See Also

Informat:
- “NLMNLEEKw.d Informat” on page 498

NLMNIEGPw.d Informat
Reads the monetary format of the international expression for Egypt.

Category: Numeric
Alignment: Left

Syntax
NLMNIEGPw.d

Syntax Description

\( w \)
specifies the width of the output field.

Default 9
Range 1–32

\( d \)
specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\( x=\text{input}('($12,345.67')',\text{nlmniegp32.2}); \)
\( y=\text{input}('($12,345.67')',\text{dollar32.2}); \)
Statements | Results
---|---
put x=;  | -12345.67
put y=;  | -12345.67

See Also

Informat:

- “NLMNLEGW.d Informat” on page 499

**NLMNIEURw.d Informat**

Reads the monetary format of the international expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

Category: Numeric
Alignment: Left

**Syntax**

NLMNIEURw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{input}'(\$12,345.67)',\text{nlmnieur32.2});
\]
\[
y = \text{input}'(\$12,345.67)'\text{dollar32.2});
\]
### Statements | Results
---+---+1----+
put x=; | -12345.67
put y=; | -12345.67

#### See Also

**Format:**

- “NLMNIEURw.d Format” on page 187

---

**NLMNIGBPw.d Informat**

Reads the monetary format of the international expression for the United Kingdom.

**Category:** Numeric

**Alignment:** Left

#### Syntax

\[ \text{NLMNIGBP}w.d \]

#### Syntax Description

**\(w\)**

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

**\(d\)**

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default:** 0
- **Range:** 0–31

#### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{input} ('$12,345.67', \text{nlmnigbp32.2}); \\
y=\text{input} ('$12,345.67' \text{dollar32.2});
\]
### Statements

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

Format:
- “NLMNIGBPw.d Format” on page 188

---

**NLMNIHKDw.d Informat**

Reads the monetary format of the international expression for Hong Kong.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[ \text{NLMNIHKD} \_w.d \]

### Syntax Description

- **\( w \)** specifies the width of the output field.
  - Default: 9  
  - Range: 1–32
- **\( d \)** specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.
  - Default: 0  
  - Range: 0–31

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)',nlmnhkd32.2);
y=input'($12,345.67)'dollar32.2);
```
Statements | Results
---|---
|  
|  
| put x=; | -12345.67
| put y=; | -12345.67

### See Also

**Format:**
- “NLMNIHKDw.d Format” on page 189

---

**NLMNIHRKw.d Informat**

Reads the monetary format of the international expression for Croatia.

**Category:** Numeric

**Alignment:** Left

**Syntax**

\[ \text{NLMNIHRKw.d} \]

**Syntax Description**

\[ w \]

- Specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\[ d \]

- Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default:** 0
- **Range:** 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{input}('($12,345.67')', \text{nlmnihrk32.2}); \\
y &= \text{input}('($12,345.67')', \text{dollar32.2}); \\
\end{align*}
\]
Statements | Results
---+----1----+
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
• “NLMNLHRKw.d Informat” on page 503

NLMNIHUFw.d Informat
Reads the monetary format of the international expression for Hungary.

Category: Numeric
Alignment: Left

Syntax
NLMNIHUFw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{input}(\'\$(12,345.67)\',\text{nlnihuf32.2});
y=\text{input}(\'\$(12,345.67)\',\text{dollar32.2});
\]
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td>-12345.67</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “NLMNLHUFw.d Informat” on page 504

---

**NLMNIIDRw.d Informat**

Reads the monetary format of the international expression for Indonesia.

**Category:** Numeric

**Alignment:** Left

#### Syntax

`NLMNIIDRw.d`

#### Syntax Description

- **w**  
  specifies the width of the output field.
  
  Default: 9  
  Range: 1–32

- **d**  
  specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  
  Default: 0  
  Range: 0–31

#### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmniidr32.2);
y=input('($12,345.67)',dollar32.2);
```
See Also

Informats:

- “NLMNLIDRw.d Informat” on page 505

NLMNIILSw.d Informat

Reads the monetary format of the international expression for Israel.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLMNIILSw.d

Syntax Description

\(w\)

specifies the width of the output field.

- Default: 9
- Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- Default: 0
- Range: 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[x = \text{input}'(\$12,345.67)', \text{nlniils32.2);}\]
\[y = \text{input}'(\$12,345.67)'\text{dollar32.2);}\]
### Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

**See Also**

**Format:**
- “NLMNIILSw.d Format” on page 193

---

**NLMNIINRw.d Informat**

Reads the monetary format of the international expression for India.

**Category:** Numeric

**Alignment:** Left

**Syntax**

`NLMNIINRw.d`

**Syntax Description**

- `w`
  - specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- `d`
  - specifies to divide the number by $10^d$. If the data contains decimal points, the `d` value is ignored.
  - **Default:** 0
  - **Range:** 0–31

**Example**

In the following example, the `LOCALE=` system option is set to English_UnitedStates.

```plaintext
x=input('
($12,345.67)',nlmniinr32.2);  
y=input('
($12,345.67)',dollar32.2);
```
See Also

Informat:
- “NLMNLINRw.d Informat” on page 507

NLMNIJPYw.d Informat
Reads the monetary format of the international expression for Japan.

Category: Numeric
Alignment: Left

Syntax

NLMNIJPYw.d

Syntax Description

w
specifies the width of the output field.
Default 9
Range 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input'($12,345.67)',nlmnijpy32.2);
y=input'($12,345.67)'dollar32.2);
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**
- “NLMNIJYPw.d Format” on page 195

---

**NLMNIKRWw.d Informat**

Reads the monetary format of the international expression for South Korea.

- **Category:** Numeric
- **Alignment:** Left

**Syntax**

\[NLMNIKRWw.d\]

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - Default: 9
  - Range: 1–32

- **d**
  - Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 0
  - Range: 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)',nlmnikrw32.2);
y=input'($12,345.67)'dollar32.2);
```
See Also

Informat:

• “NLMNLKRWw.d Informat” on page 509

NLMNILTLw.d Informat

Reads the monetary format of the international expression for Lithuania.

Category: Numeric
Alignment: Left

Syntax

NLMNILTLw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{input}('($12,345.67)$', nlmnlit132.2);
y = \text{input}('($12,345.67)$', dollar32.2);
\]
Statements  |  Results
---|---
| 12345.67  
put x=;  
| 12345.67  
put y=;  
| 12345.67

See Also

Informat:
• “NLMNLTLw.d Informat” on page 510

NLMNILVLw.d Informat
Reads the monetary format of the international expression for Latvia.

Category: Numeric
Alignment: Left

Syntax

NLMNILVLw.d

Syntax Description

`w`
specifies the width of the output field.

Default  9
Range  1–32

`d`
specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.

Default  0
Range  0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=input('($12,345.67)',nlmnilv32.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>1-4-7-1</td>
</tr>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

#### Informat:
- “NLMNLLVLw.d Informat” on page 511

---

### NLMNIMOPw.d Informat

Reads the monetary format of the international expression for Macau.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

```
NLMNIMOPw.d
```

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0
  - **Range:** 0–31

#### Example

In the following example, the `LOCALE=` system option is set to English_UnitedStates.

```bash
x=input('($12,345.67)',nlnimop32.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “NLMNLMOPw.d Informat” on page 512

---

**NLMNIMXNw.d Informat**

Reads the monetary format of the international expression for Mexico.

**Category:** Numeric  
**Alignment:** Left

**Syntax**

\[ \text{NLMNIMXN}w.d \]

**Syntax Description**

- \( w \) specifies the width of the output field.
  - Default: 9  
  - Range: 1–32
- \( d \) specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 0  
  - Range: 0–31

**Example**

In the following example, the `LOCATE=` system option is set to `English_UnitedStates`.

\[ \text{x=input('$(12,345.67)$',nmlnimxn32.2);} \]
\[ \text{y=input('$(12,345.67)$',dollar32.2);} \]
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-12345.67</td>
</tr>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “NLMNLMXNw.d Informat” on page 513

---

**NLMNIMYRw.d Informat**

Reads the monetary format of the international expression for Malaysia.

- **Category:** Numeric
- **Alignment:** Left

### Syntax

```
NLMNIMYRw.d
```

### Syntax Description

- **w**
  - specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0
  - **Range:** 0–31

### Example

In the following example, the **LOCALE=** system option is set to **English_UnitedStates**.

```plaintext
x=input('($12,345.67)',nlmnimyr32.2);
y=input('($12,345.67)',dollar32.2);
```
See Also

Format:

• “NLMNIMYRw.d Format” on page 201

NLMNINOKw.d Informat

Reads the monetary format of the international expression for Norway.

Category: Numeric
Alignment: Left

Syntax

NLMNINOKw.d

Syntax Description

w
specifies the width of the output field.

Default 9
Range 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input'($12,345.67)',nlmninok32.2);
y=input'($12,345.67)dollar32.2);
### NLMNINZDw.d Informat

Reads the monetary format of the international expression for New Zealand.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

```
NLMNINZDw.d
```

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0
  - **Range:** 0–31

#### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmninzd32.2);
y=input'($12,345.67)',dollar32.2);
```
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

**Format:**

- “NLMNINZDw.d Format” on page 203

---

**NLMNIPLNw.d Informat**

Reads the monetary format of the international expression for Poland.

**Category:** Numeric

**Alignment:** Left

**Syntax**

`NLMNIPLNw.d`

**Syntax Description**

`w`

specifies the width of the output field.

- Default: 9
- Range: 1–32

`d`

specifies to divide the number by $10^d$. If the data contains decimal points, the `d` value is ignored.

- Default: 0
- Range: 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)',nlmnipln32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x=...</code></td>
<td>-12345.67</td>
</tr>
<tr>
<td><code>y=...</code></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
See Also

Format:
- “NLMNIPLNw.d Format” on page 204

NLMNIRUBw.d Informat
Reads the monetary format of the international expression for Russia.

Category: Numeric
Alignment: Left

Syntax
NLMNIRUBw.d

Syntax Description

w
specifies the width of the output field.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1–32</td>
</tr>
</tbody>
</table>

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0–31</td>
</tr>
</tbody>
</table>

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input'($12,345.67)',nlmnbrub32.2);
y=input'($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;  -12345.67</td>
<td></td>
</tr>
<tr>
<td>put y=;  -12345.67</td>
<td></td>
</tr>
</tbody>
</table>
### NLMNISEKw.d Informat

Reads the monetary format of the international expression for Sweden.

**Category:** Numeric  
**Alignment:** Left

### Syntax

\[
\text{NLMNISEKw.d}
\]

### Syntax Description

\(w\)

specifies the width of the output field.  
**Default:** 9  
**Range:** 1–32

\(d\)

specifies to divide the number by 10\(^d\). If the data contains decimal points, the \(d\) value is ignored.  
**Default:** 0  
**Range:** 0–31

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{input}'($12,345.67)'n\text{lmnisek32.2)};  
y=\text{input}'($12,345.67)'dollar32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLMNIRUBw.d Format” on page 205
NLMNISGDw.d Informat

Reads the monetary format of the international expression for Singapore.

**Category:** Numeric

**Alignment:** Left

### Syntax

\[ \text{NLMNISGD} w.d \]

### Syntax Description

- **w**
  
  specifies the width of the output field.
  
  Default: 9
  
  Range: 1–32

- **d**
  
  specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  
  Default: 0
  
  Range: 0–31

### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input'($12,345.67)','nlmnisgd32.2);
y=input'($12,345.67) 'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

**Format:**

- “NLMNISEKw.d Format” on page 206
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNISGDw.d Format” on page 207

NLMNITHBw.d Informat

Reads the monetary format of the international expression for Thailand.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNITHBw.d

**Syntax Description**

w

specifies the width of the output field.

Default 9

Range 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 0

Range 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnithb32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### NLMNITRYw.d Informat

Reads the monetary format of the international expression for Turkey.

**Category:** Numeric  
**Alignment:** Left

### Syntax

`NLMNITRYw.d`

### Syntax Description

- `w`  
  specifies the width of the output field.  
  - **Default:** 9  
  - **Range:** 1–32

- `d`  
  specifies to divide the number by $10^d$. If the data contains decimal points, the `d` value is ignored.  
  - **Default:** 0  
  - **Range:** 0–31

### Example

In the following example, the `LOCALE=` system option is set to English UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnitry32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td>-12345.67</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

- “NLMNLTHBw.d Informat” on page 521
### Statements and Results

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td>-12345.67</td>
</tr>
<tr>
<td><code>put y=;</code></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “NLMNLTRYw.d Informat” on page 522

---

### NLMNITWDw.d Informat

Reads the monetary format of the international expression for Taiwan.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

```
NLMNITWDw.d
```

#### Syntax Description

**w**
- Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

**d**
- Specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.
  - **Default:** 0
  - **Range:** 0–31

#### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input'($12,345.67)',nlmitwd32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1-----</td>
</tr>
<tr>
<td></td>
<td>-1-----</td>
</tr>
</tbody>
</table>
See Also

Format:

- “NLMNITWD.w.d Format” on page 210

NLMNIUSDw.d Informat

Reads the monetary format of the international expression for Puerto Rico and the United States.

Category: Numeric
Alignment: Left

Syntax

NLMNIUSDw.d

Syntax Description

w

specifies the width of the output field.

Default 9
Range 1–32

d

specifies to divide the number by 10\(^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{input}'($12,345.67)',\text{nlmniusd32.2});
y=\text{input}'($12,345.67)'\text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
### See Also

**Format:**
- “NLMNIUSDw.d Format” on page 211

---

**NLMNIZARw.d Informat**

Reads the monetary format of the international expression for South Africa.

**Category:** Numeric

**Alignment:** Left

**Syntax**

\[ \text{NLMNIZARw.d} \]

**Syntax Description**

- \( w \) specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- \( d \) specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.
  - **Default:** 0
  - **Range:** 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=input'($12,345.67)',nlmnizar32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
See Also

Format:

- “NLMNIZARw.d Format” on page 212

### NLMNLAEDw.d Informat

Reads the monetary format of the local expression for the United Arab Emirates.

**Category:** Numeric  
**Alignment:** Left

### Syntax

NLMNLAEDw.d

### Syntax Description

- **w**  
  Specifies the width of the output field.
  
  **Default** 9  
  **Range** 1–32

- **d**  
  Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  
  **Default** 0  
  **Range** 0–31

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmlnaed32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:

- “NLMNIAEDw.d Informat” on page 452

NLMNLAUDw.d Informat

Reads the monetary format of the local expression for Australia.

Category: Numeric
Alignment: Left

Syntax

NLMNLAUDw.d

Syntax Description

w

specifies the width of the output field.

Default 9
Range 1–32

d

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('$(12,345.67)',nlnlaud32.2);
y=input('$(12,345.67)',dollar32.2);

Statements | Results
--- | ---
 | -12345.671

---

---

---
### See Also

**Format:**
- “NLMNLAUDw.d Format” on page 214

### NLMNLBGNw.d Informat

Reads the monetary format of the local expression for Bulgaria.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

```
NLMNLBGNw.d
```

#### Syntax Description

- **$w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **$d**
  - Specifies to divide the number by 10$^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0
  - **Range:** 0–31

#### Example

In the following example, the `LOCALE=` system option is set to English_UnitedStates.

```
x=input(-12345.67,nlmnlbgn32.2);
y=input(-12345.67,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
### NLMNLBR Lw.d Informat

Reads the monetary format of the local expression for Brazil.

**Category:** Numeric

**Alignment:** Left

#### Syntax

\[
\text{NLMNLBR Lw.d}
\]

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 9
  - Range: 1–32

- **d**
  - Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 0
  - Range: 0–31

#### Example

In the following example, the LOCATE= system option is set to English_UnitedStates.

```plaintext
x=input('$(12,345.67)',nlmnlbr132.2);
y=input('$(12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**

- “NLMNIBGNw.d Informat” on page 454
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
- “NLMNBRLw.d Informat” on page 455

### NLMNLCAw.d Informat

Reads the monetary format of the local expression for Canada.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

```
NLMNLCAw.d
```

#### Syntax Description

- **w**
  - specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
  - **Default:** 0
  - **Range:** 0–31

#### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=input('($12,345.67)',nlmlcad32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---+---+---+---+---+---+---+---+---+---</td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
`put x=;` | -12345.67
`put y=;` | -12345.67

See Also

Format:

- “NLMNLCDw.d Format” on page 217

NLMNLCHFw.d Informat

Reads the monetary format of the local expression for Liechtenstein and Switzerland.

**Category:** Numeric

**Alignment:** Left

**Syntax**

\[ \text{NLMNLCHF}_w.d \]

**Syntax Description**

\( w \)

- Specifies the width of the output field.

  Default: 9

  Range: 1–32

\( d \)

- Specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

  Default: 0

  Range: 0–31

**Example**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```r
x=input('($12,345.67)',nlnlchf32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>---</code></td>
<td><code>---</code></td>
</tr>
</tbody>
</table>
### NLMNLCNYw.d Informat

Reads the monetary format of the local expression for China.

**Category:** Numeric  
**Alignment:** Left

#### Syntax

NLMNLCNY\(w.d\)

#### Syntax Description

\(w\)

- Specifies the width of the output field.  
  - **Default:** 9  
  - **Range:** 1–32

\(d\)

- Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.  
  - **Default:** 0  
  - **Range:** 0–31

#### Example

In the following example, the \texttt{LOCALE= system option} is set to English\_UnitedStates.

\[
x = \text{input}('($12,345.67')', \text{nlmnlcny32.2});
\]

\[
y = \text{input}('($12,345.67')', \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>-12345.67</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Format:  
- “NLMNLCHFw.d Format” on page 218
See Also

Format:

- “NLMNLCNYw.d Format” on page 219

### NLMNLCZKw.d Informat

Reads the monetary format of the local expression for the Czech Republic.

**Category:** Numeric

**Alignment:** Left

**Syntax**

\[ \text{NLMNLCZK}w.d \]

**Syntax Description**

- **\(w\)** specifies the width of the output field.
  
  Default: 9  
  Range: 1–32

- **\(d\)** specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  
  Default: 0  
  Range: 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{input}'(\$(12,345.67)',\text{nlmnlczk32.2});
y=\text{input}'(\$(12,345.67)',\text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

### See Also

**Informat:**

- “NLMNICZKw.d Informat” on page 459

---

### NLMNLDKKw.d Informat

Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.

**Category:** Numeric

**Alignment:** Left

---

### Syntax

\[ \text{NLMNLDKK}w.d \]

### Syntax Description

- **\( w \)** specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **\( d \)** specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - **Default:** 0
  - **Range:** 0–31

### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

\[
\begin{align*}
x &= \text{input}('($12,345.67)',\text{nlmnlk32.2}); \\
y &= \text{input}('($12,345.67)',\text{dollar32.2}); \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------</td>
</tr>
</tbody>
</table>

---
Statements | Results
---|---
`put x=;` | -12345.67
`put y=;` | -12345.67

### See Also

Format:
- “NLMNLDKKw.d Format” on page 221

---

**NLMNLEEKw.d Informat**

Reads the monetary format of the local expression for Estonia.

**Category:** Numeric

**Alignment:** Left

**Syntax**

\[ \text{NLMNLEEK}w.d \]

**Syntax Description**

\[ w \]

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\[ d \]

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default:** 0
- **Range:** 0–31

**Example**

In the following example, the `LOCALE=` system option is set to English_UnitedStates.

```plaintext
x=input('$(12,345.67)',nlnleek32.2);
y=input('$(12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x=input('$(12,345.67)',nlnleek32.2);</code></td>
<td><code>-12345.67</code></td>
</tr>
<tr>
<td><code>y=input('$(12,345.67)',dollar32.2);</code></td>
<td><code>-12345.67</code></td>
</tr>
</tbody>
</table>
See Also

Informat:
• “NLMNIEEKw.d Informat” on page 461

NLMNLEGpw.d Informat
Reads the monetary format of the local expression for Egypt.

Category: Numeric
Alignment: Left

Syntax
NLMNLEGpw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies to divide the number by 10\(^d\). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{input}('$(12,345.67)',\text{nlmlegp32.2});
y=\text{input}('$(12,345.67)',\text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---

Statements | Results
---|---------
---|-------

### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

Informat:
- “NLMNIEGPw.d Informat” on page 462

---

**NLMNLEURw.d Informat**

Reads the monetary format of the local expression for Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Portugal, Slovakia, Slovenia, and Spain.

- **Category:** Numeric
- **Alignment:** Left

#### Syntax

`NLMNLEURw.d`

#### Syntax Description

- **w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the *d* value is ignored.
  - **Default:** 0
  - **Range:** 0–31

#### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input('($12,345.67)',nlmnleur32.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

Format:
- “NLMNLEURw.d Format” on page 224

---

## NLMNLGBPw.d Informat

Reads the monetary format of the local expression for the United Kingdom.

**Category:** Numeric  
**Alignment:** Left

### Syntax

NLMNLGBPw.d

### Syntax Description

**w**  
specifies the width of the output field.

- **Default:** 9  
- **Range:** 1–32

**d**  
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

- **Default:** 0  
- **Range:** 0–31

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlgbp32.2);
y=input('($12,345.67)',dollar32.2);
```
Statements | Results
---+----+-----+
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:

- “NLMNLGBPw.d Format” on page 225

NLMNLHKDw.d Informat

Reads the monetary format of the local expression for Hong Kong.

Category: Numeric

Alignment: Left

Syntax

NLMNLHKDw.d

Syntax Description

w

specifies the width of the output field.

Default 9

Range 1–32

d

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0

Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input({'($12,345.67)'},nlmnlhd32.2);
y=input({'($12,345.67)'},dollar32.2);
Statements | Results
---|---
|----|---
| put x=; | -12345.67 |
| put y=; | -12345.67 |

See Also

Format:
- “NLMNLHDKw.d Format” on page 226

**NLMNLHRKw.d Informat**

Reads the monetary format of the local expression for Croatia.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNLHRKw.d

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 9
  - Range: 1–32

- **d**
  - specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 0
  - Range: 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```r
x=input('($12,345.67)',nlmnlhrk32.2);
y=input('($12,345.67)',dollar32.2);
```
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-1,234.57</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**

- “NLMNIHRKw.d Informat” on page 466

---

**NLMNLHUFw.d Informat**

Reads the monetary format of the local expression for Hungary.

**Category:** Numeric  
**Alignment:** Left

### Syntax

`NLMNLHUFw.d`

### Syntax Description

**w**

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

**d**

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

- **Default:** 0
- **Range:** 0–31

### Example

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```plaintext
x=input('($12,345.67)',nlmnlhuf32.2);
y=input('($12,345.67)',dollar32.2);
```
See Also

Informat:
- “NLMNIHUFw.d Informat” on page 467

NLMNLIDRw.d Informat
Reads the monetary format of the local expression for Indonesia.

Category: Numeric
Alignment: Left

Syntax

NLMNLIDRw.d

Syntax Description

w
specifies the width of the output field.

Default 9
Range 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('$(12,345.67)',nlmnlidr32.2);
y=input('$(12,345.67)',dollar32.2);
```
See Also

Informat:
- “NLMNIDRw.d Informat” on page 468

NLMNLILSw.d Informat
Reads the monetary format of the local expression for Israel.

Category: Numeric
Alignment: Left

Syntax
NLMNLILSw.d

Syntax Description

\( w \)
- specifies the width of the output field.
  
  Default 9
  Range 1–32

\( d \)
- specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.
  
  Default 0
  Range 0–31

Example
In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x= input('($12,345.67)', nlmnlila32.2);
y= input('($12,345.67)', dollar32.2);
```
NLMNLINRw.d Informat

Reads the monetary format of the local expression for India.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Left</td>
</tr>
</tbody>
</table>

Syntax

NLMNLINRw.d

Syntax Description

\( w \)

- specifies the width of the output field.

- Default: 9
- Range: 1–32

\( d \)

- specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

- Default: 0
- Range: 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlir32.2);
y=input('($12,345.67)',dollar32.2);
```

See Also

Format:

- “NLMNLSw.d Format” on page 230
### Statements

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=;</code></td>
<td><code>put y=;</code></td>
</tr>
<tr>
<td><code>-12345.67</code></td>
<td><code>-12345.67</code></td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “NLMNINRw.d Informat” on page 470

---

### NLMNLJPYW.d Informat

Reads the monetary format of the local expression for Japan.

**Category:** Numeric

**Alignment:** Left

### Syntax

**NLMNLJPYW.d**

### Syntax Description

**w**

- Specifies the width of the output field.

  - **Default:** 9
  - **Range:** 1–32

**d**

- Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

  - **Default:** 0
  - **Range:** 0–31

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnljpy32.2);
y=input('($12,345.67)',dollar32.2);
```
NLMNLKRWw.d Informat

Reads the monetary format of the local expression for South Korea.

**Category:** Numeric

**Alignment:** Left

### Syntax

**NLMNLKRWw.d**

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  
  **Default:** 9
  
  **Range:** 1–32

- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  
  **Default:** 0
  
  **Range:** 0–31

### Example

In the following example, the **LOCALE=** system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlkrw32.2);
y=input('($12,345.67)',dollar32.2);
```

---

### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Format:**

- “NLMNLJPYw.d Format” on page 232
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

Informat:
- “NLMNIKRWw.d Informat” on page 472

---

### NLMNLLTLw.d Informat

Reads the monetary format of the local expression for Lithuania.

**Category:** Numeric

**Alignment:** Left

### Syntax

\[ \text{NLMNLLTLw.d} \]

### Syntax Description

- \( w \)
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- \( d \)
  - Specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - **Default:** 0
  - **Range:** 0–31

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('$(12,345.67)', nlmnlltl132.2);
y=input('$(12,345.67)', dollar32.2);
```
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**

- “NLMNILTLw.d Informat” on page 473

### NLMNLLVLw.d Informat

Reads the monetary format of the local expression for Latvia.

<table>
<thead>
<tr>
<th>Category</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Left</td>
</tr>
</tbody>
</table>

#### Syntax

NLMNLLVLw.d

#### Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 9
  - Range: 1–32
  -
- **d**
  - specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
  - Default: 0
  - Range: 0–31

#### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnllv132.2);
y=input('($12,345.67')',dollar32.2);
```
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:

- “NLMNILVLw.d Informat” on page 474

**NLMNLMOw.d Informat**

Reads the monetary format of the local expression for Macau.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNLMOw.d

**Syntax Description**

\[ w \]

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\[ d \]

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default:** 0
- **Range:** 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlmop32.2);
y=input('($12,345.67)',dollar32.2);
```
See Also

Informat:

• “NLMNIMOPw.d Informat” on page 475

NLMNLMXNw.d Informat
Reads the monetary format of the local expression for Mexico.

Category: Numeric
Alignment: Left

Syntax
NLMNLMXNw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9

Range 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

Default 0

Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{input}('($12,345.67)',
          nlmnlmxn32.2); \]
\[ y = \text{input}('($12,345.67)',
          \text{dollar32.2}); \]
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

**Informat:**
- “NLMNIMXNw.d Informat” on page 476

---

### NLMNLMYRw.d Informat

Reads the monetary format of the local expression for Malaysia.

**Category:** Numeric

**Alignment:** Left

### Syntax

**NLMNLMYRw.d**

### Syntax Description

- **w**
  - specifies the width of the output field.
  - Default: 9
  - Range: 1–32

- **d**
  - specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 0
  - Range: 0–31

### Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnlmyr32.2);
y=input('($12,345.67)',dollar32.2);
```
See Also

Format:

- “NLMNLMYRw.d Format” on page 238

NLMNLNO Kw.d Informat

Reads the monetary format of the local expression for Norway.

Category: Numeric
Alignment: Left

Syntax

NLMNLNO Kw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9
Range 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{input}(\'$(12,345.67)\', \text{nlmnlno}32.2); \\
y &= \text{input}(\'$(12,345.67)\', \text{dollar}32.2);
\end{align*}
\]
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----+</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Format:
- “NLMNLOKw.d Format” on page 239

**NLMNLOKw.d Informat**

Reads the monetary format of the local expression for New Zealand.

**Category:** Numeric  
**Alignment:** Left

**Syntax**

NLMNLOKw.d

**Syntax Description**

* w  
  specifies the width of the output field.  
  **Default**  9  
  **Range**  1–32

* d  
  specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.  
  **Default**  0  
  **Range**  0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```haskell
x=input('($12,345.67)',nlmnlnzd32.2);
y=input('($12,345.67)',dollar32.2);
```
Statements | Results
---|---

| put x=; | -12345.67 |
| put y=; | -12345.67 |

See Also

Format:

- “NLMNLNZDw.d Format” on page 240

NLMNLPLNw.d Informat

Reads the monetary format of the local expression for Poland.

Category: Numeric

Alignment: Left

Syntax

NLMNLPLNw.d

Syntax Description

w

specifies the width of the output field.

| Default | 9 |
| Range | 1–32 |

d

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

| Default | 0 |
| Range | 0–31 |

Example

In the following example, the LOCATE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmlpln32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
See Also

Format:

- “NLMNLPLNw.d Format” on page 241

NLMNLRUBw.d Informat

Reads the monetary format of the local expression for Russia.

Category: Numeric
Alignment: Left

Syntax

NLMNLRUBw.d

Syntax Description

w

specifies the width of the output field.

Default 9
Range 1–32

d

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnlrub32.2);
y=input('($12,345.67)',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNLRUBw.d Format” on page 242

NLMNLSEKw.d Informat
Reads the monetary format of the local expression for Sweden.

Category: Numeric  
Alignment: Left

Syntax

NLMNLSEKw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default 9  
Range 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default 0  
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{input}(\"($12,345.67)\", \text{nlmnlsek32.2});
y=\text{input}(\"($12,345.67)\", \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

NLMNLSGD\textit{w.d} Informat

Reads the monetary format of the local expression for Singapore.

\begin{itemize}
\item **Category:** Numeric
\item **Alignment:** Left
\end{itemize}

**Syntax**

\texttt{NLMNLSGD\textit{w.d}}

**Syntax Description**

\textit{w}

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\textit{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \textit{d} value is ignored.

- **Default:** 0
- **Range:** 0–31

**Example**

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
x=input('($12,345.67)',nlmnlsgd32.2);
y=input('($12,345.67)',dollar32.2);
\end{verbatim}

\begin{tabular}{|l|}
\hline
Statements & Results \\
\hline
\texttt{put x=;} & \texttt{-12345.67} \\
\texttt{put y=;} & \texttt{-12345.67} \\
\hline
\end{tabular}
See Also

Format:

- “NLMNLSGDw.d Format” on page 244

**NLMNLTHBw.d Informat**

Reads the monetary format of the local expression for Thailand.

**Category:** Numeric

**Alignment:** Left

**Syntax**

NLMNLTHBw.d

**Syntax Description**

**w**

specifies the width of the output field.

Default 9

Range 1–32

**d**

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default 0

Range 0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=input('($12,345.67)',nlmnlthb32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
- “NLMNITHBw.d Informat” on page 484

NLMNLTRYw.d Informat
Reads the monetary format of the local expression for Turkey.

Category: Numeric
Alignment: Left

Syntax

NLMNLTRYw.d

Syntax Description

w
specifies the width of the output field.

Default 9
Range 1–32

d
specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default 0
Range 0–31

Example

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input('($12,345.67)',nlmnltr32.2);
y=input('($12,345.67)',dollar32.2);

Statements | Results
---|---
---+---1----+
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Informat:
- “NLMNITRYw.d Informat” on page 485

**NLMNLTWDw.d Informat**

Reads the monetary format of the local expression for Taiwan.

**category:** Numeric

**Alignment:** Left

**Syntax**

NLMNLTWDw.d

**Syntax Description**

- **w**
  
  specifies the width of the output field.
  
  Default  9
  
  Range  1–32

- **d**
  
  specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
  
  Default  0
  
  Range  0–31

**Example**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnltwd32.2);
y=input('($12,345.67)',dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| | -12345.67-1-
Statements | Results
--- | ---
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNLTWDw.d Format” on page 247

NLMNUSDw.d Informat

Reads the monetary format of the local expression for Puerto Rico, and the United States.

**Category:** Numeric
**Alignment:** Left

**Syntax**

\[ \text{NLMNUSD}_{w,d} \]

**Syntax Description**

\( w \)

specifies the width of the output field.

\[\begin{align*}
\text{Default} & : 9 \\
\text{Range} & : 1\text{–}32
\end{align*}\]

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

\[\begin{align*}
\text{Default} & : 0 \\
\text{Range} & : 0\text{–}31
\end{align*}\]

**Example**

In the following example, the \textit{LOCALE=} system option is set to English United States.

\begin{verbatim}
x=input('($12,345.67)',nlmnlusd32.2);
y=input'($12,345.67)',dollar32.2);
\end{verbatim}

---+----1----+
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

---
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNLUSDw.d Format” on page 248

NLMNLZARw.d Informat

Reads the monetary format of the local expression for South Africa.

Category: Numeric
Alignment: Left

Syntax

NLMNLZARw.d

Syntax Description

\textit{w}

specifies the width of the output field.

Default 9
Range 1–32

\textit{d}

specifies to divide the number by 10^d. If the data contains decimal points, the \textit{d} value is ignored.

Default 0
Range 0–31

Example

In the following example, the \texttt{LOCALE=} system option is set to English_UnitedStates.

\begin{verbatim}
x=input('($12,345.67)',nlmnlzar32.2);
y=input'($12,345.67)',dollar32.2);
\end{verbatim}

Statements | Results
---|---
--- | ----
Statements | Results
--- | ---
put x=; | -12345.67
put y=; | -12345.67

See Also

Format:
- “NLMNLZARw.d Format” on page 249

NLMNYw.d Informat
Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

NLMNYw.d

**Syntax Description**

- **w** specifies the width of the input field.
  - Default: 9
  - Range: 1–32

- **d** specifies whether to divide the number by 10^d. If the data contains decimal separators, the d value is ignored.
  - Default: 0
  - Range: 0–31

**Details**

The NLMNYw.d informat reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the close parenthesis from the input data.

**Comparisons**

The NLMNYw.d informat performs processing that is the opposite of the NLMNYIw.d informat.

The NLMNYw.d informat is similar to the DOLLARw.d informat except that the NLMNYw.d informat is locale-specific.
NLMLNY returns an error message if you enclose numerical data with apostrophes. The apostrophes specify that the data is character. The following example produces an error message because the numerical value 1 is enclosed in apostrophes.

data;
x=input("'1'",NLMLNY32.);
put x=;
run;

Example

The following examples use the input value of $12,345.67.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>x=input('($12,345.67)',nlmny32.2);</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=input('($12,345.67)',dollar32.2);</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “NLMLNYwd Format” on page 250
- “NLMLNYIwd Format” on page 251

Informat:
- “NLMLNYIwd Informat” on page 527

NLMLNYIwd Informat

Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.

Category: Numeric

Syntax

NLMLNYIwd

Syntax Description

w

specifies the width of the input field.
Default 9  
Range 1–32

d  
specifies whether to divide the number by 10^d. If the data contains decimal  
separators, the d value is ignored.  
Default 0  
Range 0–31

Details  
The NLMNYIw.d informat reads monetary data in the specified locale for the  
international expression, and then converts the data to a numeric value. It removes any  
thousands separators, decimal separators, blanks, the currency symbol, and the close  
parenthesis from the input data.

Comparisons  
The NLMNYIw.d informat performs processing that is the opposite of the NLMNYw.d  
informat.

Example  
The following examples use the input value of 12,345.67.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>x=input(‘(USD12,345.67)’,nlmnyi32.2);</td>
<td>-12345.67</td>
</tr>
<tr>
<td>y=input(‘$-12,345.67’,dollar32.2);</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
• “NLMNYw.d Format” on page 250
• “NLMNYIw.d Format” on page 251

Informat:
• “NLMNYw.d Informat” on page 526
**NLNUMw.d Informat**

Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

```nl
NLNUMw.d
```

**Syntax Description**

- `w`
  
  Specifies the width of the input field.
  
  **Default:** 6
  
  **Range:** 1–32

- `d`
  
  Specifies whether to divide the number by \(10^d\). If the data contains decimal separators, the `d` value is ignored.
  
  **Default:** 0
  
  **Range:** 0–31

**Details**

The `NLNUMw.d` informat reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the close parenthesis from the input data.

`NLNUM` returns an error message if you enclose numerical data with apostrophes. The apostrophes specify that the data is character. The following example produces an error message because the numeric value 1 is enclosed in apostrophes:

```nl
data;
x=input('"1"',NLNUM32.);
put x=;
run;
```

**Comparisons**

The `NLNUMw.d` informat performs processing that is opposite to the `NLNUMIw.d` informat.

**Example**

The following example uses \(-123456.78\) as the input value.
Statements

Results

```
options locale=English_UnitedStates;
x=input('-1,234,356.78',nlnum32.2);
put x=;
```

-1234356.78

See Also

Formats:

- “NLNUMw.d Format” on page 252
- “NLMNYIw.d Format” on page 251

Informat:

- “NLNUMIw.d Informat” on page 530

### NLNUMIw.d Informat

Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

`NLNUMIw.d`

**Syntax Description**

\( w \)

specifies the width of the input field.

- Default: 6
- Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.

- Default: 0
- Range: 0–31

**Details**

The NLNUMIw.d informat reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value. It removes any thousands...
separators, decimal separators, blanks, the currency symbol, and the close parenthesis from the input data.

**Comparisons**

The NLNUMIw.d informat performs processing that is opposite of the NLNUMw.d informat.

**Example**

The following example uses –1,234,356.78 as the input value.

```
Statements                  Results
----+----1----+
options locale=English_UnitedStates;     -1234356.78
x=input('-1,234,356.78', nlnumi32.2);
put x=;
```

**See Also**

**Formats:**

- “NLNUMw.d Format” on page 252
- “NLNUMIw.d Format” on page 254
- “NLNUMw.d Informat” on page 529

---

**NLPCTw.d Informat**

Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

```
NLPCTw.d
```

**Syntax Description**

\( w \)

specifies the width of the input field.

- **Default**: 6
- **Range**: 1–32
\(d\)

specifies whether to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.

Default 0

Range \(0–31\)

**Details**

The NLPCT\(w.d\) informat reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value. It divides the value by 100 and removes any thousands separators, decimal separators, blanks, the percent sign, and the close parenthesis from the input data.

**Comparisons**

The NLPCT\(w.d\) informat performs processing that is opposite of the NLPCTI\(w.d\) informat. The NLPCT\(w.d\) informat is similar to the PERCENT\(w.d\) informat except that the NLPCT\(w.d\) informat is locale-specific.

NLPCT returns an error message if you enclose numerical data with apostrophes. The apostrophes specify that the data is character. The following example produces an error message because the numerical value 1 is enclosed in apostrophes.

data;
  x=input('1',NLPCT32.);
  put x=;
run;

**Example**

The following example uses \(-12,345.67\%\) as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>(-123.4567)</td>
</tr>
<tr>
<td>x=input('(''-12,345.67%',nlpct32.);</td>
<td>(-123.4567)</td>
</tr>
<tr>
<td>y=input('(''{12,345.67%}',percent32.);</td>
<td>(-123.4567)</td>
</tr>
<tr>
<td>put x=;</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

- “NLPCT\(w.d\) Format” on page 255
- “NLPCTI\(w.d\) Format” on page 256
- “NLPCTI\(w.d\) Informat” on page 533
**NLPCTI\textit{w.d} Informat**

Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

\texttt{NLPCTI\textit{w.d}}

**Syntax Description**

\texttt{w}

specifies the width of the input field.

Default: 6
Range: 1–32

\texttt{d}

specifies whether to divide the number by $10^d$. If the data contains decimal separators, the $d$ value is ignored.

Default: 0
Range: 0–31

**Details**

The NLPCTI\textit{w.d} informat reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value. It divides the value by 100 and removes any thousands separators, decimal separators, blanks, the percent sign, and the close parentheses from the input data.

**Comparisons**

The NLPCTI\textit{w.d} informat performs processing that is opposite of the NLPCT\textit{w.d} informat.

**Example**

The following example uses -12,345.67\% as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>-12,345.67%</td>
<td>-53.37</td>
</tr>
</tbody>
</table>
Statements | Results
--- | ---
.options LOCALE=English.UnitedStates; | -123.4567
x=input('-12,345.67%',nlpct32.2); | -123.4567
y=input('{12,345.67%},percent32.2); | 
put x=; | 
put y=; | 

See Also

Formats:
- “NLPCTw.d Format” on page 255
- “NLPCTIw.d Format” on page 256

Informat:
- “NLPCTw.d Informat” on page 531

NLTIMAPw. Informat

Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value.

Category: Date and Time

Syntax

NLTIMAPw.

Syntax Description

w
specifies the width of the input field.

Default 10

Range 4–200

Example

The following example uses 04:24:43 p.m. as the input value.

Statements | Results
--- | ---
---+----1----+
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>16:24:43</td>
</tr>
<tr>
<td>y=input('04:24:43 PM',nltimap11.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>16:24:00</td>
</tr>
<tr>
<td>y=input('16.24 Uhr',nltimap11.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Format:**
- “NLTIMAPw. Format” on page 264

### NLTIMEW. Informat

Reads the time value in the specified locale, and then converts the time value to the local SAS time value.

**Category:** Date and Time  
**Alias:** NLTIMAP

### Syntax

`NLTIMEW.`

### Syntax Description

<table>
<thead>
<tr>
<th><code>w</code></th>
<th>specifies the width of the input field.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>10–200</td>
</tr>
</tbody>
</table>

### Example

The following example uses 16:24:43 as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>16:24:43</td>
</tr>
<tr>
<td>y=input('16:24:43',nltime.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
</tbody>
</table>
Statements

```latex
options locale=German_Germany;
y=input('16.24 Uhr',nltime.);
put y time.;
```

Results

<table>
<thead>
<tr>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=German_Germany;</td>
</tr>
<tr>
<td>y=input('16.24 Uhr',nltime.);</td>
</tr>
<tr>
<td>put y time.;</td>
</tr>
</tbody>
</table>

See Also

Format:

- “NLTIMEw. Format” on page 265

$\text{REVERJw. Informat}$

Reads character data from right to left and preserves blanks.

Category: Character

Syntax

$\text{REVERJw:}$

Syntax Description

<table>
<thead>
<tr>
<th>$w$</th>
<th>specifies the width of the input field.</th>
</tr>
</thead>
</table>

Default 1 if $w$ is not specified

Range 1–32767

Comparisons

The $\text{REVERJw.}$ informat is similar to the $\text{REVERS}w$. informat except that $\text{REVERS}w$. informat left aligns the result by removing all leading blanks.

Example

The following example uses ABCD as the input value.

```
input @1 name $reverj7.;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>#DCBA</td>
</tr>
</tbody>
</table>
The character # represents a blank space.

**See Also**

**Informat:**
- “$REVERSw. Informat” on page 537

### $REVERSw. Informat

Reads character data from right to left, and then left aligns the text.

**Category:** Character

#### Syntax

\$REVERSw.

#### Syntax Description

\( w \)

- specifies the width of the input field.

**Default** 1 if \( w \) is not specified

**Range** 1–32767

#### Comparisons

The $REVERSw. informat is similar to the $REVERJw. informat except that $REVERJw. informat preserves all leading and trailing blanks.

#### Example

The following example uses ABCD as the input value.

```
input @1 name $revers7.;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>DCBA###</td>
</tr>
</tbody>
</table>

* The character # represents a blank space.
### Values

| ABCD | DCBA### |
| * | * |

* The # character represents a blank space.

### See Also

**Informat:**
- “$REVERJw. Informat” on page 536

---

### $UCS2Bw. Informat

Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

**Category:** Character

#### Syntax

$UCS2Bw:

#### Syntax Description

**w**

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

- **Default:** 8
- **Range:** 2–32000

#### Comparisons

The $UCS2Bw. informat performs processing that is opposite of the $UCS2BEw. informat. If you are processing data within the same operating environment, then use the $UCS2Xw. informat. If you are processing data from different operating environments, then use the $UCS2Bw. and $UCS2Lw. informats.

#### Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
</table>
Statements | Result
--- | ---
x=input('5927'x,$ucs2b.); | x=91e5
put x=$hex4.;

See Also

Formats:
- “$UCS2Bw. Format” on page 266
- “$UCS2Lw. Format” on page 269
- “$UCS2Xw. Format” on page 271
- “$UTF8Xw. Format” on page 288

Informats:
- “$UCS2Lw. Informat” on page 540
- “$UCS2Xw. Informat” on page 542
- “$UTF8Xw. Informat” on page 557

$UCS2BEw. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to big-endian, 16-bit, UCS2, Unicode encoding.

Category: Character

Syntax

$UCS2BEw.

Syntax Description

w

w specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8

Range 1–32000

Comparisons

The $UCS2BEw. informat performs processing that is opposite of the $UCS2Bw. informat.
**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2020</td>
</tr>
<tr>
<td>\texttt{ucs2str=input ('া', }\texttt{ $ucs2be2.)};}</td>
<td></td>
</tr>
<tr>
<td>\texttt{put ucs2str=$hex4.;}</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**
- “$UCS2Bw. Format” on page 266
- “$UCS2BEw. Format” on page 268

**Informat:**
- “$UCS2Bw. Informat” on page 538

---

**$UCS2Lw. Informat**

Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

**Category:** Character

**Syntax**

\texttt{$\text{UCS2L}w$}

**Syntax Description**

\texttt{w}

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

<table>
<thead>
<tr>
<th>Default</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>2–32000</td>
</tr>
</tbody>
</table>

**Comparisons**

The $\text{UCS2L}w$. informat performs processing that is opposite of the $\text{UCS2LE}w$. informat. If you are processing data within the same operating environment, then use the $\text{UCS2X}w$.informat. If you are processing data from different operating environments, then use the $\text{UCS2B}w$. and $\text{UCS2L}w$. informats.
Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('2759'x,$ucs2l.');</td>
<td>x=91e5</td>
</tr>
<tr>
<td>put x=$hex4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Bw. Format” on page 266
- “$UCS2Lw. Format” on page 269
- “$UCS2Xw. Format” on page 271
- “$UTF8Xw. Format” on page 288

Informats:
- “$UCS2Bw. Informat” on page 538
- “$UCS2Xw. Informat” on page 542
- “$UTF8Xw. Informat” on page 557

$UCS2LEw. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to little-endian, 16-bit, UCS2, Unicode encoding.

Category: Character

Syntax

$UCS2LEw.

Syntax Description

\w

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8

Range 1–32000
Comparisons

The $UCS2LEw. informat performs processing that is opposite of the $UCS2Lw. informat.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input (' salari', $ ucs2le2.); put ucs2str=$hex4;</td>
<td>ucs2str=2759</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Lw. Format” on page 269
- “$UCS2LEw. Format” on page 270

Informat:
- “$UCS2Lw. Informat” on page 540

$UCS2Xw. Informat

Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

**Category:** Character

**Syntax**

$UCS2Xw.

**Syntax Description**

$w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2–32000</td>
</tr>
</tbody>
</table>
Comparisons

The $UCS2Xw. informat performs processing that is the opposite of the $UCS2XEw. informat. If you are processing data within the same operating environment, then use the $UCS2Xw. informat. If you are processing data from different operating environments, then use the $UCS2Bw. and $UCS2Lw. informats.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment. This example uses little-endian formatting.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('5927'x,$ucs2x.); put x=$hex4.;</td>
<td>x=91e5</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Bw. Format” on page 266
- “$UCS2Lw. Format” on page 269
- “$UCS2Xw. Format” on page 271
- “$UTF8Xw. Format” on page 288

Informats:
- “$UCS2Bw. Informat” on page 538
- “$UCS2Lw. Informat” on page 540
- “$UTF8Xw. Informat” on page 557

$UCS2XEw. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to 16-bit, UCS2, Unicode encoding.

Category: Character

Syntax

$UCS2XEw.
Syntax Description

\( w \)

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default 8
Range 1-32000

Comparisons

The \( $\text{UCS2XE}w \). informat performs processing that is opposite of the \( $\text{UCS2X}w \). informat.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ucs2str=input ('%', $ucs2xe2.);</code></td>
<td><code>ucs2str=5927</code></td>
</tr>
<tr>
<td><code>put ucs2str=$hex6;</code></td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “\( $\text{UCS2X}w \). Format” on page 271
- “\( $\text{UCS2XE}w \). Format” on page 273

Informat:
- “\( $\text{UCS2X}w \). Informat” on page 542

\( $\text{UCS4B}w \). Informat

Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

Category: Character

Syntax

\( $\text{UCS4B}w \).
**Syntax Description**

\[ w \]

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4

Range: 4–32000

**Comparisons**

If you are processing data within the same operating environment, then use the \$UCS4Xw. informat. If you are processing data from different operating environments, then use the \$UCS4Bw. and \$UCS4Lw. informats.

**Example**

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>z=put('Zero1',$UCS4B20.);</td>
<td>Zero1</td>
</tr>
<tr>
<td>x=input(z,$UCS4B20.);</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Format:

- “\$UCS4Bw. Format” on page 274

Informats:

- “\$UCS4Lw. Informat” on page 545
- “\$UCS4Xw. Informat” on page 546

---

**\$UCS4Lw. Informat**

Reads a character string that is encoded in little-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

**Category:** Character

**Syntax**

\$UCS4Lw.
**Syntax Description**

\[ w \]

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default 4

Range 4–32000

**Comparisons**

If you are processing data within the same operating environment, then use the \$UCS4Xw. informat. If you are processing data from different operating environments, then use the \$UCS4Bw. and \$UCS4Lw. informats.

**Example**

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>z=put('.com',$UCS4L16.); put z $hex32.;</td>
<td>2E000000630000006F0000006D000000</td>
</tr>
</tbody>
</table>

**See Also**

Format:
- “\$UCS4Lw. Format” on page 276

Informats:
- “\$UCS4Bw. Informat” on page 544
- “\$UCS4Xw. Informat” on page 546

**\$UCS4Xw. Informat**

Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.

**Category:** Character

**Syntax**

\$UCS4Xw.
Syntax Description

\( w \)

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4  
Range: 4–32000

Comparisons

The $UCS4Xw$ informat performs processing that is the opposite of the $UCS4XEw$ informat. Use the $UCS4Xw$ informat when you are processing data within the same operating environment. Use the $UCS4Bw$ and $UCS4Lw$ informats when you are processing data from different operating environments.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment. This example uses little-endian formatting.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs4=put(’91e5’x,$ucs4x.);</td>
<td>ucs4=27590000</td>
</tr>
<tr>
<td>sjis=input(ucs4,$ucs4x.);</td>
<td>sjis=91E52020</td>
</tr>
<tr>
<td>put ucs4=$hex8. sjis=$hex8.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Xw. Format” on page 271
- “$UCS2Bw. Format” on page 266
- “$UCS2Lw. Format” on page 269
- “$UCS4Xw. Format” on page 279
- “$UTF8Xw. Format” on page 288

Informats:
- “$UCS2Bw. Informat” on page 538
- “$UCS2Lw. Informat” on page 540
- “$UTF8Xw. Informat” on page 557
$UCS4XEw. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to 32-bit, UCS4, Unicode encoding.

**Category:** Character

**Syntax**

$UCS4XEw.

**Syntax Description**

*W*

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

<table>
<thead>
<tr>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1–32000</td>
</tr>
</tbody>
</table>

**Comparisons**

The $UCS4XEw. informat performs processing that is the opposite of the $UCS4Xw. informat.

**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs4str=input ('_execution', $ucs4xe2.);</td>
<td>ucs4str=00005927</td>
</tr>
<tr>
<td>put ucs4str=$hex8;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

- “$UCS4Xw. Format” on page 279
- “$UCS4XEw. Format” on page 280

**Informat:**

- “$UCS4Xw. Informat” on page 546
$UESCw. Informat
Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session.

Category: Character

Syntax
$UESCw.

Syntax Description

\textit{w}

specifies the width of the output field.

Default 8

Range 1–32000

Details
If the characters are not available on all operating environments (for example, 0–9, a–z, A–Z) they must be represented in UESC representation. The $UESCw. informat can be nested.

Comparisons
The $UESCw. informat performs processing that is the opposite of the $UESCEw. informat.

Example
These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('¥u5927', $uesc10.); y=input('¥uu5927', $uesc10.); z=input('¥uuu5927', $uesc10.); put x; put y; put z;</td>
<td>¥u5927 ¥uu5927</td>
</tr>
</tbody>
</table>

See Also

Formats:
$UESCEw. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UESC representation.

**Category:** Character

**Syntax**

$UESCEw.

**Syntax Description**

\( w \)

specifies the width of the input field.

**Default** 8

**Range** 1–32000

**Details**
The $UESCEw. informat can be nested.

**Comparisons**
The $UESCEw. informat performs processing that is opposite of the $UESCw. informat.

**Example**

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = \text{input('¥u5927', $uesc10.)}; )</td>
<td>¥u5927</td>
</tr>
<tr>
<td>( y = \text{input('¥uu5927', $uesc10.)}; )</td>
<td>¥uu5927</td>
</tr>
<tr>
<td>( z = \text{input('¥uuu5927', $uesc10.)}; )</td>
<td>¥uuu5927</td>
</tr>
<tr>
<td>put x y z;</td>
<td></td>
</tr>
</tbody>
</table>
$UNCRw. Informat

Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.

Category: Character

Syntax

$UNCRw.

Syntax Description

w

specifies the width of the input field.

Default 8

Range 1–32000

Details

The input string must contain only characters and NCR. Any national characters must be represented in NCR.

Comparisons

The $UNCRw. informat performs processing that is opposite of the $UNCREw. informat.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
</table>
| ---+-----+1----+
Statements | Result
--- | ---
x=input ('&amp;#22823;','$uncr10.); | ¥
y=input('abc','$uncr10.); | abc
put X;
put Y;

See Also

Formats:
- “$UNCRw. Format” on page 284
- “$UNCREw. Format” on page 285

Informat:
- “$UNCREw. Informat” on page 552

$UNCREw. Informat

Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.

Category: Character

Syntax

$UNCREw:

Syntax Description

w

specifies the width of the input field.

Default 8

Range 1–32000

Details

The output string converts to plain characters and NCR. Any national characters convert to NCR.

Comparisons

The $UNCREw. informat performs processing that is the opposite of the $UNCRw. informat.
Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('abc', $uncre12.); put x;</td>
<td>$#22823;abc</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UNCRw. Format” on page 284
- “$UNCREw. Format” on page 285

Informat:
- “$UNCRw. Informat” on page 551

$UPARENw. Informat

Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session.

**Category:** Character

**Syntax**

$UPARENw.

**Syntax Description**

w

specifies the width of the input field.

Default 8
Range 1–32000

**Details**

If the SAS session encoding does not have a corresponding Unicode expression, the expression remains in encoding of the current SAS session.
Comparisons

The $UPARENw. informat performs processing that is opposite of the $UPARENEw. informat.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('&lt;u0061&gt;',$uparen10.);</td>
<td>a</td>
</tr>
<tr>
<td>w=input('&lt;u0062&gt;',$uparen10.);</td>
<td>b</td>
</tr>
<tr>
<td>x=input('&lt;u0063&gt;',$uparen10.);</td>
<td>c</td>
</tr>
<tr>
<td>y=input('&lt;u0033&gt;',$uparen10.);</td>
<td>3</td>
</tr>
<tr>
<td>z=input('&lt;u5927&gt;',$uparen10.);</td>
<td>$</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UPARENEw. Format” on page 286
- “$UPARENEw. Format” on page 287

Informats:
- “$UPARENEw. Informat” on page 554
- “$UPARENPw. Informat” on page 555

$UPARENEw. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UPAREN representation.

Category: Character

Syntax

SUPARENEw.

Syntax Description

$w$

specifies the width of the input field.
Default: 8

Range: 1–32000

Comparisons

The $UPARENEw. informat performs processing that is opposite of the $UPARENw. informat.

Example

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('a',$uparen10.);</td>
<td>&lt;u0061&gt;</td>
</tr>
<tr>
<td>w=input('b',$uparen10.);</td>
<td>&lt;u0062&gt;</td>
</tr>
<tr>
<td>x=input('c',$uparen10.);</td>
<td>&lt;u0063&gt;</td>
</tr>
<tr>
<td>y=input('3',$uparen10.);</td>
<td>&lt;u0033&gt;</td>
</tr>
<tr>
<td>z=input('מנות','$uparen10.);</td>
<td>&lt;u5927&gt;</td>
</tr>
</tbody>
</table>

put v;
put w;
put x;
put y;
put z;

See Also

Formats:

- “$UPARENw. Format” on page 286
- “$UPARENEw. Format” on page 287

Informats:

- “$UPARENw. Informat” on page 553
- “$UPARENPw. Informat” on page 555

$UPARENPw. Informat

Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session, with national characters that remain in the encoding of the UPAREN representation.

Category: Character
**Syntax**

$\text{UPAREN}w$.

**Syntax Description**

$w$ specifies the width of the input field.

- **Default**: 8
- **Range**: 1–32000

**Details**

If the UPAREN expression contains a national character, whose value is greater than Unicode 0x00ff, the expression remains as a UPAREN expression.

**Example**

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('{&lt;u0061&gt;',$uparen10.);</td>
<td>a</td>
</tr>
<tr>
<td>w=input('{&lt;u0062&gt;',$uparen10.);</td>
<td>b</td>
</tr>
<tr>
<td>x=input('{&lt;u0063&gt;',$uparen10.);</td>
<td>c</td>
</tr>
<tr>
<td>y=input('{&lt;u0033&gt;',$uparen10.);</td>
<td>3</td>
</tr>
<tr>
<td>z=input('{&lt;u5927&gt;',$uparen10.);</td>
<td>&lt;u5927&gt;</td>
</tr>
</tbody>
</table>

| put v;                   |
| put w;                   |
| put x;                   |
| put y;                   |
| put z;                   |

**See Also**

**Formats:**
- “$\text{SUPAREN}w. Format” on page 286
- “$\text{SUPARENE}w. Format” on page 287

**Informats:**
- “$\text{SUPAREN}w. Informat” on page 553
- “$\text{SUPARENE}w. Informat” on page 554
$UTF8Xw. Informat

 Reads a character string that is encoded in UTF-8, and then converts the character string to the encoding of the current SAS session.

 Category: Character

 Syntax

 $UTF8Xw.

 Syntax Description

 w

 specifies the width of the input field.

 Default 8

 Range 1–32000

 Comparisons

 This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

 Statements | Result
---+----+---
 x=input (' 554a7' x, $utf8x3.);  x
 put x;

 See Also

 Formats:

 - “$UCS2Bw. Format” on page 266
 - “$UCS2Lw. Format” on page 269
 - “$UCS2Xw. Format” on page 271
 - “$UTF8Xw. Format” on page 288

 Informats:

 - “$UCS2Bw. Informat” on page 538
 - “$UCS2Lw. Informat” on page 540
 - “$UCS2Xw. Informat” on page 542
$VSLOGw. Informat

Reads a character string that is in visual order, and then converts the character string to left-to-right logical order.

Category: BIDI Text Handling

Syntax

$VSLOGw.

Syntax Description

\( w \)

Specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $VSLOGw. informat performs processing that is opposite of the $VSLOGRw. informat.

Example

The following example uses the Hebrew input value of "_flight".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('_flight',$vslog12.); put x;</td>
<td>flight</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of " \( \text{ذات} \) computer."

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input (&quot; ( \text{ذات} ) computer&quot;,$vslog12.); put x;</td>
<td>computer</td>
</tr>
</tbody>
</table>
See Also

**Formats:**
- “$VSLOGRw. Format” on page 291
- “$VSLOGw. Format” on page 290

**Informat:**
- “$VSLOGRw. Informat” on page 559

### $VSLOGRw. Informat

Reads a character string that is in visual order, and then converts the character string to right-to-left logical order.

**Category:** BIDI Text Handling

#### Syntax

$VSLOGRw.$

**Syntax Description**

$w$

specifies the width of the input field.

- **Default:** 200
- **Range:** 1–32000

#### Comparisons

The $VSLOGRw.$ informat performs processing that is opposite of the $VSLOGw.$ informat.

#### Example

The following example uses the Hebrew input value of “ﬂight.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input (’תְנוֹן ﬂight’,$vsl ogr12.);</td>
<td>ﬂight</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “computer.”
<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('computer',$vslogr12.);</td>
<td>computer</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**
- “$VSLOGw. Format” on page 290
- “$VSLOGRw. Format” on page 291

**Informat:**
- “$VSLOGw. Informat” on page 558

---

**YENw.d Informat**

Removes embedded yen signs, commas, and decimal points.

**Category:** Numeric

**Syntax**

YENw.d

**Syntax Description**

w
- specifies the width of the input field.
  - Default 1
  - Range 1–32

d
- specifies the power of 10 by which to divide the value.
  - Requirement d must be 0 or 2
  - Tip If the d is 2, then YENw.d reads a decimal point and two decimal digits. If d is 0, YENw.d reads the value without a decimal part.

**Details**

The hexadecimal representation of the code for the yen sign character is 5B on EBCDIC systems and 5C on ASCII systems. The monetary character that these codes represent might be different in other countries.
Example

The following example uses yen as the input.

input value yen10.2;

<table>
<thead>
<tr>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>¥1254.71</td>
<td>1254.71</td>
</tr>
</tbody>
</table>

See Also

Format:

- “YENw.d Format” on page 297
Chapter 14
Dictionary of Macro Functions for NLS

Macro Functions by Category

The following table provides brief descriptions of the SAS NLS macro functions. For more information, see the NLS entry for each macro function.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCS</td>
<td>%KCMPRES and %QKCMPRES Macro Function (p. 566)</td>
<td>Compresses multiple blanks and removes leading and trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>%KINDEX Macro Function (p. 566)</td>
<td>Returns the position of the first character of a string.</td>
</tr>
<tr>
<td></td>
<td>%KLEFT and %QKLEFT Macro Functions (p. 567)</td>
<td>Left-aligns an argument by removing leading blanks.</td>
</tr>
<tr>
<td></td>
<td>%KLENGTH Macro Function (p. 567)</td>
<td>Returns the length of a string.</td>
</tr>
<tr>
<td></td>
<td>%KSCAN and %QKSCAN Functions (p. 568)</td>
<td>Search for a word that is specified by its position in a string.</td>
</tr>
<tr>
<td></td>
<td>%KSUBSTR and %QKSUBSTR Macro Functions (p. 570)</td>
<td>Produce a substring of a character string.</td>
</tr>
</tbody>
</table>
Dictionary

%KCMPRES and %QKCMPRES Macro Function

Compresses multiple blanks and removes leading and trailing blanks.

**Category:** DBCS  
**Type:** NLS macro function

**Syntax**

%KCMPRES (text | text expression)  
%QKCMPRES (text | text expression)

**Details**

The %KCMPRES and %QKCMPRES macro functions compress multiple blanks and removes leading and trailing blanks. %KCMPRES returns an unquoted result, even if the argument is quoted. %QKCMPRES returns a quoted result.

%QKCMPRES produces a result with the following special characters and mnemonic operators masked, so the macro processor interprets them as text instead of as elements of the macro language:

& % ' * ( ) + - / < > = ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN

%KINDEX Macro Function

Returns the position of the first character of a string.

**Category:** DBCS  
**Type:** NLS macro function

**Syntax**

%KINDEX (source, string)

**Required Arguments**

source  
  is a character string or text expression.
string

is a character string or text expression.

Details

The %KINDEX function searches source for the first occurrence of string and returns the position of its first character. If string is not found, the function returns 0.

Example: Locating a Character

The following statements find the first character V in a string:

```sas
%let a=a very long value;
%let b=%kindex(&a,v);
%put V appears at position &b..;
```

When these statements execute, the following line is written to the SAS log:

V appears at position 3.

%KLEFT and %QKLEFT Macro Functions

Left-aligns an argument by removing leading blanks.

**Category:** DBCS

**Requirement:** MAUTOSOURCE system option

**Syntax**

- `%KLEFT (text | text expression)`
- `%QKLEFT (text | text expression)`

**Details**

The %KLEFT and %QKLEFT macro functions left-align arguments by removing leading blanks. If the argument contains a special character or mnemonic operator, listed here, use QKLEFT.

%KLEFT returns an unquoted result, even if the argument is quoted. %QKLEFT produces a result with the following special characters and mnemonic operators masked so that the macro processor interprets them as text instead of as elements of the macro language:

```sas
& % ' " ( ) + - * / < > = ¬ ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN
```

%KLENGTH Macro Function

Returns the length of a string.

**Category:** DBCS

**Type:** NLS macro function
%KLENGTH (character string | text expression)

Details
If the argument is a character string, %KLENGTH returns the length of the string. If the argument is a text expression, %KLENGTH returns the length of the resolved value. If the argument has a null value, %KLENGTH returns 0.

Example: Returning String Lengths
The following statements find the lengths of character strings and text expressions:

```sas
%let a=Happy;
%let b=Birthday;
%put The length of &a is %klength(&a).;
%put The length of &b is %klength(&b).;
%put The length of &a &b To You is %klength(&a &b to you).;
```

When these statements execute, the following is written to the SAS log:

The length of Happy is 5.
The length of Birthday is 8.
The length of Happy Birthday To You is 21.

%KSCAN and %QKSCAN Functions
Search for a word that is specified by its position in a string.

<table>
<thead>
<tr>
<th>Category:</th>
<th>DBCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>NLS macro function</td>
</tr>
</tbody>
</table>

Syntax

%KSCAN (argument, n<.delimiters >)
%QKSCAN (argument, n<.delimiters >)

Required Arguments

**argument**

is a character string or a text expression. If argument contains a special character or mnemonic operator, listed here, use %QKSCAN.

**n**

is an integer or a text expression that yields an integer, which specifies the position of the word to return. If n is greater than the number of words in argument, the functions return a null string. If n is negative, %KSCAN examines the character string and selects the word that starts at the end of the string and searches backward.

**delimiter**

specifies a character variable that produces characters that you want %QKSCAN to use as word separators in the character expression.
Details

The %KSCAN and %QKSCAN functions search argument and return the nth word. A word is one or more characters separated by one or more delimiters.

%KSCAN does not mask special characters or mnemonic operators in its results, even when the argument was previously masked by a macro quoting function. %QKSCAN masks the following special characters and mnemonic operators in its results:

& % ' " ( ) + − * / < > = ¬ ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN

In the %KSCAN function, word refers to a substring that has all of the following characteristics:

- is bounded on the left by a delimiter or the beginning of the string
- is bounded on the right by a delimiter or the end of the string
- contains no delimiters

A word can have a length of zero if there are delimiters at the beginning or end of the string or if the string contains two or more consecutive delimiters.

If you use the %KSCAN function with only two arguments, then the default delimiters depend on whether your computer uses ASCII or EBCDIC characters:

- If your computer uses ASCII characters, then the default delimiters are as follows:
  blank ! $ % & ( ) * + , . / ; < ^ |
  In ASCII environments that do not contain the ^ character, the %KSCAN function uses the ~ character instead.
- If your computer uses EBCDIC characters, then the default delimiters are as follows:
  blank ! $ % & ( ) * + , . / ; < ¬ | ¢¦

The %KSCAN function allows character arguments to be null. Null arguments are treated as character strings with a length of zero. Numeric arguments cannot be null.

Example: Comparing the Actions of %KSCAN and %QKSCAN

This example illustrates the actions of %KSCAN and %QKSCAN:

```%macro a;
   aaaaaa
%mend a;
%macro b;
   bbbbbbb
%mend b;
%macro c;
   ccccccc
%mend c;
%let x=%nrstr(%a*%b*%c);
%put X: &x;
%put The third word in X, with KSCAN: %kscan(&x,3,*);
%put The third word in X, with QKSCAN: %qkscan(&x,3,*);
```

The %PUT statement writes these lines to the log:

```
X: %a*%b*%c
The third word in X, with KSCAN: ccccccc
```
%KSUBSTR and %QKSUBSTR Macro Functions

Produce a substring of a character string.

Category: DBCS
Type: NLS macro function

Syntax

%KSUBSTR (argument, position<, length> )
%QKSUBSTR (argument, position<, length> )

Required Arguments

argument
is a character string or a text expression. If argument contains a special character or mnemonic operator, listed here, use %QKSUBSTR.

position
is an integer or an expression (text, logical, or arithmetic) that yields an integer that specifies the position of the first character in the substring. If position is greater than the number of characters in the string, %KSUBSTR and %QKSUBSTR issue a warning message and return a null value.

length
is an optional integer or an expression (text, logical, or arithmetic) that yields an integer that specifies the number of characters in the substring. If length is greater than the number of characters following position in argument, %KSUBSTR and %QKSUBSTR issue a warning message and return a substring containing the characters from position to the end of the string. By default, %KSUBSTR and %QKSUBSTR produce a string containing the characters from position to the end of the character string.

Details

The %KSUBSTR and %QKSUBSTR functions produce a substring of argument, which begins at position and continues for the number of characters in length.

%KSUBSTR does not mask special characters or mnemonic operators in its result.

%QKSUBSTR masks the following special characters and mnemonic operators:

\& % ' " ( ) + - * / < > = ¬ ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN

Examples

Example 1: Limiting a Fileref to Eight Characters

The macro MAKEFREF uses %KSUBSTR to assign the first eight characters of a parameter as a fileref, in case a user assigns one that is longer:

```latex
%macro makefref(fileref,file);
  %if %klength(&fileref) gt 8 %then
```
Example 2: Storing a Long Macro Variable Value in Segments

The macro SEPMSG separates the value of the macro variable MSG into 40-character units and stores each unit in a separate variable:

```sas
%macro sepmsg(msg);
  %let i=1;
  %let start=1;
  %if %length(&msg)>40 %then
    %do;
      %do %until(%klength(&&msg&i)<40);
        %let msg&i=%qksubstr(&msg,&start,40);
        %put Message &i is: &&msg&i;
        %let i=%eval(&i+1);
        %let start=%eval(&start+40);
        %let msg&i=%qksubstr(&msg,&start);
      %end;
      %put Message &i is: &&msg&i;
    %end;
  %else %put No subdivision was needed.;
%mend sepmsg;
```

When this program executes, these lines are written to the SAS log:

Message 1 is: A character operand was found in the %EV
Message 2 is: AL function or %IF condition where a nu
Message 3 is: meric operand is required. A character
Message 4 is: operand was found in the %EVAL function
Message 5 is: or %IF condition where a numeric operand is required.

Example 3: Comparing the Actions of %KSUBSTR and %QKSUBSTR

%KSUBSTR produces a resolved result because it does not mask special characters and mnemonic operators in the C language before processing it:

```sas
%let a=one;
%let b=two;
%let c=%nrstr(&a &b);
%put C: &c;
%put With KSUBSTR: %ksubstr(&c,1,2);
%put With QKSUBSTR: %qKsubstr(&c,1,2);
```

When these statements execute, these lines are written to the SAS log:

C: &a &b
With KSUBSTR: one
%KUPCASE and %QKUPCASE Macro Functions

Convert values to uppercase.

Category: DBCS
Type: NLS macro function

Syntax

%KUPCASE (character string | text expression)

%QKUPCASE (character string | text expression)

Details

The %KUPCASE and %QKUPCASE functions convert lowercase characters in the argument to uppercase. %KUPCASE does not mask special characters or mnemonic operators in its results.

If the argument contains a special character or mnemonic operator, listed here, use %QKUPCASE. %QKUPCASE masks the following special characters and mnemonic operators in its results:

& % ' " ( ) + − * / < > = ¬ ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN

%KUPCASE and %QKUPCASE are useful in comparing values because the macro facility does not automatically convert lowercase characters to uppercase before comparing them.

Examples

Example 1: Capitalizing a Value to Be Compared

In this example, the macro RUNREPT compares a value input for the macro variable MONTH to the string DEC. If the uppercase value of the response is DEC, then PROC FSVIEW runs on the data set REPORTS.ENDYEAR. Otherwise, PROC FSVIEW runs on the data set with the name of the month in the REPORTS data library.

```
%macro runrept(month);
  %if %kupcase(&month)=DEC %then
    %str(proc fsview data=reports.endyear; run;);
  %else %str(proc fsview data=reports.&month; run;);
%mend runrept;
```

You can invoke the macro in any of these ways to satisfy the %IF condition:

```
%runrept(DEC)
%runrept(Dec)
%runrept(dec)
```

Example 2: Comparing %KUPCASE and %QKUPCASE

These statements show the results produced by %KUPCASE and %QKUPCASE:

```
572  Chapter 14  •  Dictionary of Macro Functions for NLS
With QKSUBSTR: &a
```
%let a=begin;
%let b=%nrstr(&a);
%put KUPCASE produces: %kupcase(&b);
%put QKUPCASE produces: %qkupcase(&b);

When these statements execute, the following is written to the SAS log:

KUPCASE produces: BEGIN
QKUPCASE produces: &A
Part 8

System Options for NLS

Chapter 15

Dictionary of System Options for NLS .......................... 577
System Option Entries by Category

The language control category of SAS system options are affected by NLS. The following table provides brief descriptions of the SAS system options. For more detailed descriptions, see the dictionary entry for each SAS system option:
<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment Control: Language Control</td>
<td>DATESY&lt;style=&gt;le System Option (p. 580)</td>
<td>Specifies the sequence of month, day, and year when ANYD&lt;style&gt;TDTE, ANYD&lt;style&gt;TDM, or ANYD&lt;style&gt;TME informat data is ambiguous.</td>
</tr>
<tr>
<td></td>
<td>DBCS System Option: UNIX, Windows, and z/OS (p. 581)</td>
<td>Recognizes double-byte character sets (DBCS).</td>
</tr>
<tr>
<td></td>
<td>DBCSlang System Option: UNIX, Windows, and z/OS (p. 582)</td>
<td>Specifies a double-byte character set (DBCS) language.</td>
</tr>
<tr>
<td></td>
<td>DBCS TY style System Option: UNIX, Windows, and z/OS (p. 583)</td>
<td>Specifies the encoding method to use for a double-byte character set (DBCS).</td>
</tr>
<tr>
<td></td>
<td>DFLANG System Option: UNIX, Windows, and z/OS (p. 585)</td>
<td>Specifies the language for international date informats and formats.</td>
</tr>
<tr>
<td></td>
<td>ENCODING System Option: UNIX, Windows, and z/OS (p. 587)</td>
<td>Specifies the default character-set encoding for the SAS session.</td>
</tr>
<tr>
<td></td>
<td>FSDBTYPE System Option: UNIX (p. 589)</td>
<td>Specifies a full-screen double-byte character set (DBCS) encoding method.</td>
</tr>
<tr>
<td></td>
<td>FSIMM System Option: UNIX (p. 590)</td>
<td>Specifies input method modules (IMMs) for full-screen double-byte character set (DBCS).</td>
</tr>
<tr>
<td></td>
<td>FSIMMOPT System Option: UNIX (p. 591)</td>
<td>Specifies options for input method modules (IMMs) that are used with a full-screen double-byte character set (DBCS).</td>
</tr>
<tr>
<td></td>
<td>LOCALE System Option (p. 591)</td>
<td>Specifies a set of attributes in a SAS session that reflect the language, local conventions, and culture for a geographical region.</td>
</tr>
<tr>
<td></td>
<td>LOCALEDATA System Option: UNIX, Windows, and z/OS (p. 593)</td>
<td>Specifies the source database for the locale information.</td>
</tr>
<tr>
<td></td>
<td>LOGLANGENG System Option (p. 595)</td>
<td>Specifies using the English language for SAS log message text when the LOCALE option is set after start-up.</td>
</tr>
<tr>
<td></td>
<td>MAPEBCDIC2ASCII System Option (p. 597)</td>
<td>Specifies a translation table that SAS uses to transcode from EBCDIC to ASCII and from ASCII to EBCDIC.</td>
</tr>
<tr>
<td></td>
<td>ODSLANGCHG System Option (p. 599)</td>
<td>Determines whether the language of the text of the ODS output can be changed.</td>
</tr>
<tr>
<td></td>
<td>TIMEZONE System Option (p. 604)</td>
<td>Specifies the user local time zone.</td>
</tr>
<tr>
<td></td>
<td>TRANTAB System Option (p. 605)</td>
<td>Specifies the translation tables that are used by various parts of SAS.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>URLENCODING= System Option (p. 612)</td>
<td>Specifies whether the argument to the URLENCODE function and to the URLDECODE function is interpreted using the SAS session encoding or UTF-8 encoding.</td>
</tr>
<tr>
<td>Files: External Files</td>
<td>BOMFILE System Option (p. 579)</td>
<td>Specifies whether to write the byte-order mark (BOM) prefix on Unicode-encoded external files.</td>
</tr>
<tr>
<td></td>
<td>RSASIOTRANSENTER Error System Option (p. 601)</td>
<td>Displays a transcoding error when invalid data is read from a remote application.</td>
</tr>
<tr>
<td></td>
<td>VALIDMEMNAME= System Option (p. 607)</td>
<td>Specifies the rules for naming SAS data sets, SAS data views, and item stores.</td>
</tr>
<tr>
<td></td>
<td>VALIDVARNAME= System Option (p. 610)</td>
<td>Specifies the rules for valid SAS variable names that can be created and processed during a SAS session.</td>
</tr>
<tr>
<td>Input Control: Data Processing</td>
<td>DATESTYLE= System Option (p. 580)</td>
<td>Specifies the sequence of month, day, and year when ANYDTDTE, ANYDTDTM, or ANYDTTME informat data is ambiguous.</td>
</tr>
<tr>
<td>Language Control</td>
<td>LSWLANG System Option (p. 596)</td>
<td>Specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation.</td>
</tr>
<tr>
<td>Log and Procedure Output</td>
<td>PAGESIZE= System Option (p. 600)</td>
<td>Specifies the number of lines that compose a page of the SAS log and SAS output.</td>
</tr>
<tr>
<td>Log and Procedure Output</td>
<td>PAGESIZE= System Option (p. 600)</td>
<td>Specifies the number of lines that compose a page of the SAS log and SAS output.</td>
</tr>
<tr>
<td>Log and Procedure Output</td>
<td>PAGESIZE= System Option (p. 600)</td>
<td>Specifies the number of lines that compose a page of the SAS log and SAS output.</td>
</tr>
<tr>
<td>Sort: Procedure Options</td>
<td>SORTSEQ= System Option: UNIX, Windows, and z/OS (p. 602)</td>
<td>Specifies a language-specific collating sequence for the SORT and SQL procedures to use in the current SAS session.</td>
</tr>
</tbody>
</table>

**Dictionary**

**BOMFILE System Option**

Specifies whether to write the byte-order mark (BOM) prefix on Unicode-encoded external files.

**Valid in:** configuration file, SAS invocation, OPTIONS statement, SAS System Options window
PROC OPTIONS

Group= EXTFILES

Syntax

BOMFILE | NOBOMFILE

Syntax Description

BOMFILE
Specifies to write a byte-order mark (BOM) prefix when a Unicode-encoded file is written to an external file.

NOBOMFILE
Specifies not to write a BOM prefix when a Unicode-encoded file is written to an external file.

Details

The BOMFILE system option does not apply when a Unicode-encoded external file is read.
A BOM is a signature at the beginning of a Unicode data stream. The size of the BOM varies depending on the encoding.

DATESTYLE= System Option

Specifies the sequence of month, day, and year when ANYDSTDTE, ANYDSTDTM, or ANYDTTME informat data is ambiguous.

Valid in: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Categories: Environment Control: Language Control
Input Control: Data Processing

Default: The default value is determined by the value of the LOCALE= system option.

Note: This option cannot be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

Syntax

DATESTYLE= MDY | YMD | DMY | LOCALE

Syntax Description

MDY
specifies that SAS set the order as month, day, year.

YMD
specifies that SAS set the order as year, month, day.
DMY
 specifies that SAS set the order as day, month, year.

LOCALE
 specifies that SAS set the order based on the value that corresponds to the
 LOCALE= system option value and is one of the following: MDY | YMD | DMY.

Details
 System option DATESTYLE= identifies the order of month, day, and year. The default
 value is LOCALE. The default LOCALE system option value is English. Therefore, the
 default DATESTYLE order is MDY.

To get the default settings for each locale option value, see Locale Values on page 697.

See Also

Informs:
  • “ANYDTDTEw. Informat” in SAS Formats and Informats: Reference
  • “ANYDTDTMw. Informat” in SAS Formats and Informats: Reference
  • “ANYDTTMEw. Informat” in SAS Formats and Informats: Reference

System Options:
  • “LOCALE System Option” on page 591

---

**DBCS System Option: UNIX, Windows, and z/OS**

Recognizes double-byte character sets (DBCS).

**Valid in:** configuration file, SAS invocation  
**Category:** Environment Control: Language Control

**PROC OPTIONS**

**GROUP=** LANGUAGECONTROL

**Default:** NODBCS

**UNIX specifics:** Also valid in SASV9_OPTIONS environment variable

**Syntax**

- `DBCS` | `-NODBCS` (UNIX and Windows)  
- `DBCS` | `NODBCS` (z/OS)

**Required Arguments**

**DBCS**
 recognizes double-byte character sets (DBCS) for encoding values. DBCS encodings
 are used to support East Asian languages.
NODBCS
does not recognize a DBCS for encoding values. Instead, a single-byte character set (SBCS) is used for encoding values. A single byte is used to represent each character in the character set.

Details
The DBCS system option is used for supporting languages from East Asian countries such as Chinese, Japanese, Korean, and Taiwanese.

See Also
Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 37
- “DBCS Values for a SAS Session” on page 711
- Chapter 22, “Encoding Values in SAS Language Elements,” on page 713

System Options:
- “DBCSLANG System Option: UNIX, Windows, and z/OS” on page 582
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 583

DBCSLANG System Option: UNIX, Windows, and z/OS
Specifies a double-byte character set (DBCS) language.

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS
GROUP= LANGUAGECONTROL
Default: none
UNIX specifics: Also valid in SASV9_OPTIONS environment variable

Syntax
-DBCSLANG language (UNIX and Windows)
DBCSLANG = language ( z/OS)

Required Argument
language
depends on the operating environment. The following table contains valid language values:
Table 15.1  Supported DBCS Languages According to Operating Environment

<table>
<thead>
<tr>
<th>Language</th>
<th>z/OS</th>
<th>UNIX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHINESE</td>
<td>yes*</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>(simplified)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JAPANESE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>KOREAN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TAIWANESE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>(traditional)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NONE</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

* For z/OS only, HANGUL is a valid alias for KOREAN and HANZI is a valid alias for CHINESE.

Details

The proper setting for the DBCSLANG system option depends on which setting is used for the DBCSTYPE system option. Some of the settings of DBCSTYPE support all of the DBCSLANG languages. Other settings of DBCSTYPE support only Japanese.

CHINESE specifies the language used in the People's Republic of China, which is known as simplified Chinese. TAIWANESE specifies the Chinese language used in Taiwan, which is known as traditional Chinese.

See Also

- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 37
- “DBCS Values for a SAS Session” on page 711
- Chapter 22, “Encoding Values in SAS Language Elements,” on page 713

System Options:

- “DBCS System Option: UNIX, Windows, and z/OS” on page 581
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 583

**DBCSTYPE System Option: UNIX, Windows, and z/OS**

Specifies the encoding method to use for a double-byte character set (DBCS).

- **Valid in:** configuration file, SAS invocation
- **Category:** Environment Control: Language Control
- **PROC OPTIONS GROUP=** LANGUAGECONTROL
- **z/OS specifics:** IBM
UNIX specifics:
Depends on the specific machine
Also valid in SAS9_OPTIONS environment variable

Windows specifics:
PCMS

Syntax

- `DBCSTYPE encoding-method` (UNIX and Windows)

`DBCSTYPE = encoding-method` (z/OS)

Required Argument

`encoding-method`

specifies the method that is used to encode a double-byte character set (DBCS).

Valid values for `encoding-method` depend on the standard that the computer hardware manufacturer applies to the operating environment.

Details

DBCS encoding methods vary according to the computer hardware manufacturer and the standards organization.

The DBCSLANG= system option specifies the language that the encoding method is applied to. You should specify DBCSTYPE= only if you also specify the DBCS and DBCSLANG= system options.

z/OS DBCSTYPE= supports the DBCSTYPE= value of IBM.

Comparisons

Table 15.2  DBCS Encoding Methods for z/OS

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>IBM EBCDIC encoding method</td>
</tr>
</tbody>
</table>

Table 15.3  DBCS Encoding Methods for UNIX

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>DEC encoding method</td>
</tr>
<tr>
<td>EUC</td>
<td>Extended UNIX Code encoding method</td>
</tr>
<tr>
<td>HP15</td>
<td>Hewlett Packard encoding method</td>
</tr>
<tr>
<td>PCIBM</td>
<td>IBM PC encoding method</td>
</tr>
<tr>
<td>PCMS</td>
<td>Microsoft PC encoding method</td>
</tr>
<tr>
<td>SJIS</td>
<td>Shift-JIS encoding method for the Japanese language only</td>
</tr>
<tr>
<td>DBCSTYPE= Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>NONE</td>
<td>Disables DBCS processing</td>
</tr>
</tbody>
</table>

Table 15.4  
<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCMS</td>
<td>Microsoft PC encoding method</td>
</tr>
<tr>
<td>WINDOWS</td>
<td>Alias for PCMS</td>
</tr>
<tr>
<td>SJIS</td>
<td>Shift-JIS encoding method for the Japanese language only</td>
</tr>
</tbody>
</table>

See Also

Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 37
- “DBCS Values for a SAS Session” on page 711
- Chapter 22, “Encoding Values in SAS Language Elements,” on page 713

System Options:
- “DBCS System Option: UNIX, Windows, and z/OS” on page 581
- “DBCSSLANG System Option: UNIX, Windows, and z/OS” on page 582

DFLANG= System Option: UNIX, Windows, and z/OS
Specifies the language for international date informats and formats.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment Control: Language Control

PROC OPTIONS GROUP=
LANGUAGECONTROL

Default: English

Syntax

DFLANG='language', locale

Syntax Description

'language'
specifies the language that is used for international date informats and formats.
These languages are valid values for `language`:

- Afrikaans
- Catalan
- Croatian
- Czech
- Danish
- Dutch
- English
- Finnish
- French
- German
- Hungarian
- Italian
- Japanese
- Macedonian
- Norwegian
- Polish
- Portuguese
- Russian
- Slovenian
- Spanish
- Swedish
- Swiss_French
- Swiss_German

`locale`

the locale that is specified with the locale system option becomes the active locale.

**Details**

You can change the value of the DFLANG system option during a SAS session, but you can use only one language at a time. The values for `language` are not case-sensitive.

When you specify `DFLANG=locale`, the locale that is specified in the system option of the LOCALE statement becomes the active locale. The locale or language must be supported by the DFLANG system option.

In the following example, the international date informats and formats would be German. The posix name for the German locale is de_DE. The German locale is supported by the DFLANG system option.

```sas
option locale=de_DE; /* German locale */
option DFLANG=locale;
```

In the following example, the international date informats and formats would be English. Maltese is not supported by DFLANG, so the default locale is English.

```sas
option locale=mt_MT; /* Maltese locale */
```
When you specify `DFLANG=locale`, the output of the date format is displayed in the locale that is specified with the `LOCALE` system option. To control the date format in the output, the `DFLANG` locale uses the value based on the `LOCALE` system option that has been set at start-up. If `DFLANG` is set to a valid language, then the date format in the output is English by default. In the following example, the locale is set to French and a listing output is specified:

```
Sas.exe -locale French
  Proc print data=sashelp.class ; run ;
```

```
mercredi 09 mars 2011 14 h 25
```

If you set `DFLANG=JAPANESE`, then `DFLANG` behaves the same way as `DFLANG=LOCALE`.

In SAS 9.2 the use of the `DFLANG=` option was extended. The value `JAPANESE` was added to allow customers running the Japanese image to see the datetime stamp in the SAS Log and the Listing formatted for Japanese. The value `LOCALE` was added later to give all customers the same feature.

- If `DFLANG` is not set at start up, the option value is set by the `LOCALE` option.
- The default format of the datetime stamp of the SAS log is English. If `DFLANG` is set to `LOCALE` or `JAPANESE`, the datetime stamp in the SAS Log is formatted according to the current SAS LOCALE.
- The default format of the datetime stamp of the SAS listing is English. If `DFLANG` is set to `LOCALE` or `JAPANESE`, the datetime stamp in each page of the output sent to the SAS Listing or ODS destinations is formatted according to the SAS LOCALE that was in effect immediately after `DFLANG=LOCALE` is set.
- Changes to the `LOCALE` option do not impact how the datetime stamp is formatted unless the `DTRESET` option is enabled. Then the datetime stamp is formatted according to the current SAS LOCALE.

In SAS 9.2, 9.3, and 9.4, you can use the value `JAPANESE` to process the Japanese image to see the datetime stamp in the SAS log and the listing formatted for Japanese. The value `LOCALE` is added in later releases.

### See Also

`DTRESET` System Option

---

**ENCODING System Option: UNIX, Windows, and z/OS**

Specifies the default character-set encoding for the SAS session.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>configuration file, SAS invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Environment Control: Language Control</td>
</tr>
<tr>
<td>PROC OPTIONS</td>
<td>LANGUAGECONTROL</td>
</tr>
<tr>
<td>GROUP=</td>
<td>OpenVMS specifics: latin1</td>
</tr>
<tr>
<td></td>
<td>z/OS specifics: OPEN_ED-1047</td>
</tr>
</tbody>
</table>
Windows specifics: wlatin1

Syntax

-ENCODING= ASCIIANY | EBCDICANY | encoding-value (UNIX and Windows)
ENCODEING= encoding-value (UNIX, Windows, and z/OS)

Required Arguments

ASCIIANY
Transcoding normally occurs when SAS detects that the session encoding and data set encoding are different. ASCIIANY enables you to create a data set that SAS will not transcode if the SAS session that accesses the data set has a session that encoding value of ASCII. If you transfer the data set to a machine that uses EBCDIC encoding, transcoding occurs.

Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

EBCDICANY
is valid only for z/OS. Transcoding normally occurs when SAS detects that the session encoding and the data set encoding are different. EBCDICANY enables you to create a data set that SAS does not transcode if the SAS session accessing the data set has a session encoding value of EBCDIC. If you transfer the data set to a machine that uses ASCII encoding, transcoding occurs.

encoding-value
For valid values for all operating environments, see Chapter 23, “Encoding Values for a SAS Session,” on page 723.

Details

A character-set encoding is a set of characters that have been mapped to numeric values called code points.

The encoding for a SAS session is determined by the values of the ENCODING=, LOCALE=, DBCSTYPE=, and DBCSLANG= system options as follows:

• If the ENCODING option is not specified, the value of Encoding is determined by the value of LOCALE and the operating system where SAS is running. Also, if LOCALE is not set, the default LOCALE is en_US.

• If both LOCALE= and ENCODING= are specified, the session encoding is the value that is specified by the ENCODING= option.

• If LOCALE= is specified and ENCODING= is not specified, SAS infers the appropriate encoding value from the LOCALE= value.

• If the DBCS option is set, the values for the DBCSLANG= and DBCSTYPE= system options determine the ENCODING= and LOCALE= values.

See Also

Conceptual Information:

• “Overview of Locale Concepts for NLS” on page 5
• Conceptual discussion about “Overview: Encoding for NLS” on page 9
FSDBTYPE System Option: UNIX

Specifies a full-screen double-byte character set (DBCS) encoding method.

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment Control: Language Control
PROC OPTIONS GROUP= LANGUAGECONTROL
Default: DEFAULT
UNIX specifics: all

Syntax
-FSDBTYPE encoding-method

Details
The FSDBTYPE= system option specifies the encoding method that is appropriate for a full-screen DBCS enabling method. Full-screen DBCS encoding methods vary according to the computer hardware manufacturer and the standards organization.

Table 15.5 Full-Screen DBCS Encoding Methods

<table>
<thead>
<tr>
<th>FSDBTYPE= Encoding Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dec</td>
<td>Digital Equipment Corporation encoding method</td>
</tr>
<tr>
<td>euc</td>
<td>Extended UNIX encoding method</td>
</tr>
<tr>
<td>hp15</td>
<td>HP-UX encoding method</td>
</tr>
<tr>
<td>jis7</td>
<td>7-bit Shift-JIS encoding method used in an X windows environment for the Japanese language only</td>
</tr>
<tr>
<td>pcibm</td>
<td>IBM PC encoding method</td>
</tr>
<tr>
<td>sjs</td>
<td>Shift-JIS encoding method for the Japanese language only</td>
</tr>
<tr>
<td>default</td>
<td>default method that is used by the specific host</td>
</tr>
</tbody>
</table>
FSIMM System Option: UNIX

Specifies input method modules (IMMs) for full-screen double-byte character set (DBCS).

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment Control: Language Control

PROC OPTIONS
GROUP=LANGUAGECONTROL

Default: none

UNIX specifics: all

Syntax

-\texttt{FSIMM fsdevice\_name=IMM-name1<, fsdevice\_name=IMM-name2> ...}

Details

You can specify the following values for IMM-name:

TTY | SASWUJT
provides an interface for /dev/tty. This IMM enables you to enter DBCS strings through a terminal emulator that has DBCS input capability.

PIPE | SASWUJP
provides a pipe interface. This interface forks the DBCS input server process. The default server name is saswujms, which uses the vendor-supplied MOTIF toolkit.

For example, to use the PIPE input method module for X11 drivers, you would specify:

-\texttt{FSIMM X11=PIPE}

\textit{Note:} The server is specified by using the FSIMMOPT option.

See Also

Conceptual Information:

- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 37
- “DBCS Values for a SAS Session” on page 711
- Chapter 22, “Encoding Values in SAS Language Elements,” on page 713
FSIMMOPT System Option: UNIX

Specifies options for input method modules (IMMs) that are used with a full-screen double-byte character set (DBCS).

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment Control: Language Control
PROC OPTIONS
GROUP= LANGUAGECONTROL
Default: none
UNIX specifics: all

Syntax
-FSIMMOPT fullscreen-IMM:IMM-option

Details
The FSIMMOPT system option specifies an option for each full-screen IMM (input method module). You can specify only one FSIMMOPT option for each IMM. If you specify multiple FSIMMOPT options for the same IMM, only the last specification is used.


For example, you can use the FSIMMOPT option to specify the name of the server, MOTIF, to be used for the PIPE IMM:

-.fsimmopt PIPE:MOTIF

See Also

Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 37

System Option:
- “FSIMM System Option: UNIX ” on page 590

LOCALE System Option

Specifies a set of attributes in a SAS session that reflect the language, local conventions, and culture for a geographical region.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category: Environment Control: Language Control
PROC OPTIONS
GROUP=
Default: English_UnitedStates
UNIX specifics: Also valid in SASV9_OPTIONS environment variable

Syntax

-LOCALE locale-name (UNIX and Windows)
LOCALE=locale-name (UNIX, Windows, and z/OS)

Required Argument

locale-name
For a complete list of locale values (SAS names and POSIX names), see “LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options” on page 697.

Details

The LOCALE= system option is used to specify the locale, which reflects the local conventions, language, and culture a geographical region.

If the value of the LOCALE= system option is not compatible with the value of the ENCODING= system option, the character-set encoding is determined by the value of the ENCODING= system option.

If the DBCS= system option is active, the values of the DBCSTYPE= and DBCSLANG= system options determine the locale and character-set encoding.

When you set a value for LOCALE=, the value of the following system options is modified unless explicit values have been specified:

ENCODING=
The locale that you set has a common encoding value that is used most often in the operating environment where SAS runs. If the ENCODING= option is not set explicitly in a config file or on the command line, SAS uses the ENCODING that is default for the LOCALE and operating system. The LOCALE might be set explicitly or can default. When the ENCODING= system option is set, the TRANTAB= system option is also set.

DATESTYLE=
When LOCALE= is set, the DATESTYLE= system option uses the value that corresponds to the chosen locale.

DFLANG=
When LOCALE= is set, the DFLANG= system option is set to a value that corresponds to the chosen locale.

PAPERSIZE=
When LOCALE= is set, the PAPERSIZE= system option is set to a value that corresponds to the chosen locale and the ODS printer is set to the preferred unit of measurement, inches or centimeters, for that locale.

CAUTION:
Under the Windows operating systems only: The LOCALE= option can be used to specify PAPERSIZE= only if the UNIVERSALPRINT and UPRINTMENUSWITCH system options are also specified. For details, see the
See Also

Conceptual Information:
- Chapter 2, “Locale for NLS,” on page 5
- “LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options” on page 697

System Options:
- “ENCODING System Option: UNIX, Windows, and z/OS” on page 587
- “DATESTYLE= System Option” on page 580
- “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585
- “PAPERSIZE= System Option” in SAS System Options: Reference
- “TRANTAB= System Option” on page 605

LOCALEDATA System Option: UNIX, Windows, and z/OS

Specifies the source database for the locale information.

Valid in:
- configuration file, SAS invocation

Category:
- Environment Control: Language Control

PROC OPTIONS
- LANGUAGECONTROL

Default:
- SASLOCALE

Syntax

LOCALEDATA=SASLOCALE | REGISTRY

Syntax Description

SASLOCALE
- specifies the locale definition from an internal SAS database.

REGISTRY
- specifies the locale definition from the SAS registry.

Details

The SASLOCALE value system option specifies an internal database as the source of the locale’s definition. This database does not allow customization.

The REGISTRY value specifies an external database where locale definitions are customized by using PROC LOCALEDATA.
**LOGLANGCHG System Option**

Specifies whether the language of the text output to the SAS log can be changed.

- **Valid in:** configuration file, SAS invocation
- **PROC OPTIONS GROUP= LOGCONTROL**
- **Default:** NOLOGLANGCHG

**Syntax**

```syntax
LOGLANGCHG | NOLOGLANGCHG
```

**Syntax Description**

- **LOGLANGCHG** specifies that the language of the SAS log messages can be changed after start-up.
- **NOLOGLANGCHG** specifies that the language of the SAS log message cannot be changed after start-up.

**Details**

The following conditions result in these actions:

<table>
<thead>
<tr>
<th>ODSLANGCHG</th>
<th>LOGLANGCHG</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>ON</td>
<td>Language for ODS output and the SAS log can be changed based on LSWLANG= option setting.</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>Language for ODS output and the SAS log cannot be changed.</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>Language for the SAS log can be changed based on LSWLANG= option setting.</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>Language for ODS output can be changed based on LSWLANG= option.</td>
</tr>
</tbody>
</table>

**See Also**

- “ODSLANGCHG System Option” on page 599
- “LSWLANG System Option” on page 596
- “LOGLANGENG System Option” on page 595
**LOGLANGENG System Option**

Specifies using the English language for SAS log message text when the LOCALE option is set after start-up.

- **Valid in:** configuration file, SAS invocation
- **Category:** Environment Control: Language Control
- **PROC OPTIONS GROUP=** LANGUAGECONTROL
- **Default:** NOLOGLANGENG

**Syntax**

```syntax
LOGLANGENG | NOLOGLANGENG
```

**Syntax Description**

- **LOGLANGENG**
  SAS log messages are displayed in English using the LSW feature.
- **NOLOGLANGENG**
  The LSW feature is not used to enable English for SAS log messages.

**Details**

The LOGLANGENG option interacts with the LOGLANGCHG, ODSLANGCHG, and LSWLANG options.

- **LOGLANGENG** has the same functionality as LSWLANG=EN and NOLOGLANGCHG.
- **ODSLANGCHG** is valid with LOGLANGENG.
- **NOLOGLANGENG** has no impact on the SAS session.
- During start-up, the configuration file (!sasroot\nls) and the LOCALE option determine the language for SAS messages. After start-up, if the LOCALE option is set and the LOGLANGENG option is on, the language for the SAS log output is English.
- If LOGLANGENG is enabled at start-up and the value of LOCALE is changed during the session, the SAS log output remains in English.
- If LOGLANGCHG is on, the SAS log is controlled by the LSWLANG= option.
- If ODSLANGCHG is on, the ODS text is controlled by the LSWLANG= option.
- If LOGLANGENG is not enabled at start-up and LOCALE is changed during the session, the SAS log output is displayed in the language that was set at start-up, unless both LOGLANGCHG and LSWLANG= are specified and LSWLANG has a value other than LOCALE.

**Example**

This example is a French SAS session with LOGLANGENG set to ON.
If LOGLANGENG is set to ON, then LSWLANG=EN and LOGLANGCHG is ON automatically. The SAS session always uses English for the SAS log, but the language of the ODS output is determined by the current SAS configuration.

See Also

- “LSWLANG System Option” on page 596
- “LOGLANGCHG System Option” on page 594
- “ODSLANGCHG System Option” on page 599

LSWLANG System Option

Specifies the language for the language switching feature when the LOGLANGCHG or ODSLANGCHG system option is set at SAS invocation.

Valid in: configuration file, SAS invocation

Category: Language Control

PROC OPTIONS
GROUP=LANGUAGECONTROL
Default: LOCALE

Syntax

LSWLANG=LOCALE | language

Required Arguments

LOCALE

specifying the LOCALE argument preserves the behavior prior to SAS 9.4, where the SAS message text matches the value of the LOCALE= option.

language

Specifies the language for ODS and the SAS log output. The following values can be specified:

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>English</td>
</tr>
<tr>
<td>FR</td>
<td>French</td>
</tr>
<tr>
<td>IT</td>
<td>Italian</td>
</tr>
<tr>
<td>DE</td>
<td>German</td>
</tr>
<tr>
<td>ES</td>
<td>Spanish</td>
</tr>
<tr>
<td>ZH</td>
<td>Chinese S</td>
</tr>
<tr>
<td>ZT</td>
<td>Chinese T</td>
</tr>
</tbody>
</table>
MAPEBCDIC2ASCII= System Option

Specifies a translation table that SAS uses to transcode from EBCDIC to ASCII and from ASCII to EBCDIC.

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS GROUP= LANGUAGECONTROL
Alias: MAPE2A
Interaction: The MAPEBCDIC2ASCII= system option specifies a translation table to use for the SAS session. The table specified with MAPEBCDIC2ASCII is used by components such as $EBCDIC and $ASCII formats.

Syntax

MAPEBCDIC2ASCII= TRANTAB catalog-entry

Required Argument

catalog-entry

specifies a SAS catalog TRANTAB entry that contains translation tables. If you specify entry-name.type, SAS first searches SASUSER.PROFILE, then SASHELP.LOCALE, and then SASHELP.HOST for the name specified.

Details

MAPEBCDIC2ASCII= supports the requirements of national languages.

CAUTION:

Do not change a translation table unless you are familiar with its purpose.
Translation tables are used internally by SAS to implement NLS. If you are
unfamiliar with translation tables, do not change the specifications without proper technical advice.

**NLDECSEPARATOR System Option**

Specifies whether SAS produces locale sensitive numeric output for the decimal separator or continues to format numbers with US English preferences.

**Valid in:**
configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**PROC OPTIONS GROUP=**
LANGUAGECONTROL

**Alias:**
NLD

**Default:**
NONLSDECSEPARATOR

**Syntax**

NLDECSEPARATOR | NONLDECSEPARATOR

**Required Arguments**

**NLDECSEPARATOR**
Enables locale-sensitive numeric output for the decimal separator.

**NONLDECSEPARATOR**
Disables locale-sensitive numeric output for the decimal separator.

**Details**

Use the BESTDOTX format with the NLDECSEPARATOR option to produce a numeric value with a dot as a decimal separator. If you do not use the BESTDOTX format, the result could contain a comma (1,2). SAS does not interpret this output as a numeric value. It would result in an error if the value is used in a later assignment statement.

Some SAS functions, such as CALL SYMPUT, convert numeric values to a character string using the BEST format by default. When NLDECSEPARATOR is enabled, the BEST format uses the separator character that matches the preferences for the current locale. If the locale uses a dot as a separator character, the resulting string can be used later in a SAS statement that evaluates the value as a number. However, if the separator character is a comma or other character, the result causes a syntax error if the SAS code attempts to use it as a number.

To avoid this type of error, use BESTDOTX to format the value before assigning it within the CALL SYMPUT statement. BESTDOTX always uses a dot as a decimal separator in the results that it produces. For example, in the SAS code here, BESTDOTX is used to format the value of the variable X before it is assigned to the macro variable &MACX. The value of &MACX can then be assigned to the SAS variable Y without causing a syntax error.

```sas
data a;
  x=1.2;
  CALL SYMPUT('macx', put(x, bestdotx.));
run;
```
data b;
  y=&macx
run;

See Also
“BESTDOTXw. Format” on page 104

ODSLANGCHG System Option
Determines whether the language of the text of the ODS output can be changed

Valid in: configuration file, SAS invocation
Category: Environment Control: Language Control
PROC OPTIONS GROUP=
Default: ODSLANGCHG is set to off in all servers except for the UNICODE server
Tip: The language used for the language switching feature is controlled by LSWLANG= option. By default, LSWLANG is set to LOCALE, which specifies that the language of LSW is controlled by the language of LOCALE.

Syntax

ODSLANGCHG | NOODSLANGCHG

Syntax Description

ODSLANGCHG
  Specifies that the language of ODS output can change after start-up.

NOODSLANGCHG
  Specifies that the language of ODS output cannot change after start-up.

Details

The Language Switching feature (LSW) enables you to change the language of SAS messages and ODS templates after start-up. In order to use the LSW for ODS output, you must enable ODSLANGCHG.

During start-up, the configuration file determines the language for SAS messages and ODS templates. If the ODSLANGCHG option is set, the language of ODS output can change to reflect the LSWLANG= setting when the localizations are available.

You can enable ODSLANGCHG but not translate into the language of the locale. For example, if you enable ODSLANGCHG, then start a SAS session in French and set the locale to Greek, NLDATE is displayed in Greek. The output is displayed in French. The output is displayed in French because SAS does not translate into Greek.

Comparisons

If ODSLANGCHG is enabled and LSWLANG=LOCALE is set, the ODS PATH is updated based on the LOCALE setting to include the localized templates and corresponding localized messages are used to generate ODS output.
If ODSLANGCHG is not enabled at start-up, ODS output appears in the language that was specified in the configuration file.

**Example**

Example 1 is a French server with ODSLANGCHG not enabled (NOODSLANGCHG).

If a French-client application connects to the server, the output appears in French and dates formatted by using the NL format, appear in French. If an English-client application connects to the French server, and the locale is changed to English on the server, then output messages appear in French, and dates formatted with NL formats appear in English.

Example 2 is a French server with ODSLANGCHG enabled (ODSLANGCHG) and LSWLANG=LOCALE.

If a French-client application connects to the server, the output appears in French and dates formatted by using the NL format, appear in French. If an English-client application connects to the French server, and the locale is changed to English on the server, then output messages appear in English, and dates formatted with NL format appears in English.

Example 3 is a French server with ODSLANGCHG enabled (ODSLANGCHG) and LSWLANG=English.

If a French-client application connects to the server, the output appears in English and dates formatted by using the NL format, appear in French. If an English-client application connects to the French server, and the locale is changed to English on the server, then output messages appear in English, and dates formatted with NL format appears in English.

**See Also**

- “LOGLANGENG System Option” on page 595
- “LOGLANGCHG System Option” on page 594
- “LSWLANG System Option” on page 596

**PAGESIZE= System Option**

Specifies the number of lines that compose a page of the SAS log and SAS output.

**Valid in:** Configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Categories:** Log and Procedure Output Control: SAS Log and Procedure Output
               Log and Procedure Output Control: SAS Log
               Log and Procedure Output Control: Procedure Output

**PROC OPTIONS GROUP=** LOG, LISTCONTROL
                       LISTCONTROL
                       LOGCONTROL

**Alias:** PS=

**Default:** The shipped default is 21. When SAS starts, the value is set based on the execution mode.
Note: This option cannot be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

See: “PAGESIZE System Option: UNIX” in SAS Companion for UNIX Environments
     “PAGESIZE System Option: Windows” in SAS Companion for Windows
     “PAGESIZE= System Option: z/OS” in SAS Companion for z/OS

Syntax

PAGESIZE=n | nK | hexX | MIN | MAX

Syntax Description

n | nK
  specifies the number of lines that compose a page in terms of lines (n) or units of 1,024 lines (nK).

hexX
  specifies the number of lines that compose a page as a hexadecimal number. You must specify the value beginning with a number (0–9), followed by an X. For example, the value 2dx sets the number of lines that compose a page to 45 lines.

MIN
  sets the number of lines that compose a page to the minimum setting, 15.

MAX
  sets the number of lines that compose a page to the maximum setting, 32,767.

Details

The PAGESIZE= system option affects the following output:

• the Output window for the ODS LISTING destination
• the SAS log in batch and non-interactive modes
• the ODS markup destinations when the PRINT option is used in the FILE statement in a DATA step (the FILE PRINT ODS statement is not affected by the PAGESIZE= system option)
• procedures that produce characters that cannot be scaled, such as the PLOT procedure, the CALENDAR procedure, the TIMEPLOT procedure, the FORMS procedure, and the CHART procedure

See Also


RSASIOTRANSERROR System Option

Displays a transcoding error when invalid data is read from a remote application.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES
**Syntax**

RSASIOTRANSERROR | NORSASIOTRANSERROR

**Syntax Description**

**RSASIOTRANSERROR**

specifies to display a transcoding error when invalid values are read from a remote application.

**NORSASIOTRANSERROR**

specifies not to display a transcoding error when invalid values are read from a remote application.

**Details**

The RSASIOTRANSERROR system option enables remote users of SASIO, for example, SAS Enterprise Guide and SAS Enterprise Miner, to ignore invalid data values. An invalid data value typically causes a transcoding error when the data is read by a remote application.

---

**SORTSEQ= System Option: UNIX, Windows, and z/OS**

Specifies a language-specific collating sequence for the SORT and SQL procedures to use in the current SAS session.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Sort: Procedure Options

**PROC OPTIONS GROUP=**

---

**Syntax**

SORTSEQ=collating-sequence | <LINGUISTIC>

**Syntax Description**

**collating-sequence**

specifies the collating sequence that the SORT procedure is to use in the current SAS session. Valid values can be user-supplied, or they can be one of the following:

- ASCII
- DANISH (alias NORWEGIAN)
- EBCDIC
- FINNISH
- ITALIAN
- NATIONAL
• POLISH
• REVERSE
• SPANISH
• SWEDISH

LINGUISTIC
Sorts characters according to the rules of a specified language. The rules and default collating sequence are based on the language specified in the current locale setting. The implementation is provided by the International Components for Unicode (ICU) library. Refer to “Linguistic Sorting of Data Sets and ICU” in Base SAS Procedures Guide for more information.

Details
To create or change a collating sequence, use the TRANTAB procedure to create or modify translation tables. When you create your own translation tables, they are stored in your PROFILE catalog, and they override any translation tables with the same name that are stored in the HOST catalog.

Note: System managers can modify the HOST catalog by copying newly created tables from the PROFILE catalog to the HOST catalog. All users can access the new or modified translation tables.

If you are in a windowing environment, use the Explorer window to display the SASHELP HOST catalog. In the HOST catalog, entries of type TRANTAB contain collating sequences that are identified by the entry name.

If you are not in a windowing environment, issue the following statements to generate a list of the contents of the HOST catalog. Collating sequences are entries of the type TRANTAB.

```sas
proc catalog catalog=sashelp.host;
   contents;
run;
```

To see the contents of a particular translation table, use these statements:

```sas
proc trantab table=translation-table-name;
   list;
run;
```

The contents of collating sequences are displayed in the SAS log.

Example
This example demonstrates the functionality of SORTSEQ with PROC SORT and PROC SQL:

```sas
options sortseq=reverse;
proc sort data=sashelp.class out=foo1;
   by name;
run;
proc sql;
   create table foo2 as select * from sashelp.class order by name;
quit;
run;
```

SAS provides ICU collation when the linguistic option (SORTSEQ=LINGUISTIC) is specified on the Base SAS procedure, PROC SORT. Starting in the third maintenance
release of SAS 9.4, linguistic collation can also be specified using the SORTSEQ= option in the SQL Procedure and by specifying system option SORTSEQ=LINGUISTIC.

Note: Only PROC SORT and PROC SQL are affected when the SORTSEQ=LINGUISTIC system option is specified.

When the linguistic option is specified, SAS relies on the ICU libraries as the reference implementation of the Unicode Collation Algorithm (UCA) and as a de facto standard. For in-depth information about the UCA algorithm or the International Components for Unicode (ICU) library implementation, see Download the ICU 4.8 Release and CLDR 2.0 Release Note.

See Also

- “Collating Sequence” on page 17

System Options:

- “TRANTAB= System Option” on page 605

---

**TIMEZONE= System Option**

Specifies the user local time zone.

| Valid in: | Configuration file, SAS invocation, OPTIONS statement, SAS System Options window |
| Category: | Environment Control: Language Control |
| PROC OPTIONS GROUP= | LANGUAGECONTROL |
| Alias: | TZ=, except in the restricted options configuration file where TIMEZONE= must be used |
| Note: | This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference. |

**Syntax**

```
TIMEZONE='time-zone-name' | 'time-zone-ID'
```

**Syntax Description**

- `time-zone-name`
  - specifies a three- or four-character time zone name. For example, EST is a time zone name for Eastern Time.
  - Default: BLANK, indicating that the SAS server time zone and the client time zone are the same

- `time-zone-ID`
  - For a list of time zones, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819.
time-zone-ID

specifies a region/area value that is defined by SAS. When you specify a time zone ID, the time zone that SAS uses is determined by time zone name and daylight saving time rules.

Note

Time zone IDs are compatible with Java time zone names.

See

For a list of time zone IDs, see Appendix 2, “Time Zone IDs and Time Zone Names,” on page 819.

Details

You set the TIMEZONE= option to a time zone ID or a time zone name in order for SAS to use a particular time zone. The time zone setting affects the following SAS components:

- times that are recorded by events and logs
- time of data set creation or modification
- DATE( ) function
- DATETIME( ) function
- TIME( ) function
- TODAY( ) function
- time zone functions TZONEOFF( ), TZONED( ), TZONENAME( ), TZONE2U( ), and TZONUE2S( ),
- time zone formats B8601DXw., E8601DXw., B8601LXw., E8601LXw., B8601TXw., E8601TXw., NLDATMZw., NLDATMTZw., and NLDATMWZw.

You set a time zone by specifying a time zone ID or a time zone name. A time zone ID is a region and an area separated by a forward slash (/). For example, America/New_York and Asia/Osaka are time zone IDs.

A time zone name is a three- or four-character name for a time zone. For example, EST is Eastern Time and JST is Japan Time. SAS determines the time by using time zone rules, including daylight saving time rules, before using a time value.

Some time zones names are valid for different locales. For example, CST is Central Daylight Time, Cuba Daylight Time, and China Daylight Time. SAS uses the value of the LOCALE= system option to determine the region and area to use. If TIMEZONE='CST' and LOCALE='zh_CN', SAS uses the Asia/Beijing time zone. If the time zone name does not exist for the locale, SAS searches all time zones and sets the time zone to the first match that it finds.

When this option is restricted and the value of TIMEZONE= is the default value of BLANK, time zone behavior does not use time zone information.

See Also

“Specifying Time Zones in SAS” on page 41

TRANTAB= System Option

Specifies the translation tables that are used by various parts of SAS.
The TRANTAB= system option specifies a translation table to use for the SAS session, including file transfers. The TRANTAB statement specifies a customized translation table (for example, to map an EBCDIC character to an ASCII character) to apply to the character set in the SAS file that is being exported or transferred.

**Syntax**

\[
\text{TRANTAB}=(\text{catalog-entries})
\]

**Syntax Description**

- **catalog-entries** specifies SAS catalog entries that contain translation tables. If you specify `entry-name.type`, SAS searches SASUSER.PROFILE first and then SASUSER.HOST.

**Details**

TRANTAB= was introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on the features of TRANTAB=. In SAS 9, translations tables are set by the LOCALE= system option. They are used for transcoding external files. For SAS files, there is a direct transcoding between the session encodings.

SAS 9.2 supports the TRANTAB procedure for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases.

You can list the translation tables with PROC OPTIONS. The following example is on the z/OS environment with a LOCALE option set to en_US.

\[
\text{TRANTAB}=(\text{eo}1\text{w}1\text{t}1,\text{w}1\text{t}1\text{e}0\text{l}1,\text{elat}\_\text{ucs},\text{elat}\_\text{lcs},\text{elat}\_\text{ccl},,,,\text{elat}\_\text{scc})
\]

Translation tables are specified in a list that is enclosed in parentheses and has ten positions. The position in which a table appears in the list determines the type of translation table that is specified. Individual entries in the list are separated by commas. See the list of positions and types that follows:

<table>
<thead>
<tr>
<th>Position</th>
<th>Type of Translation Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>local-to-transport-format</td>
</tr>
<tr>
<td>2nd</td>
<td>transport-to-local-format</td>
</tr>
<tr>
<td>3rd</td>
<td>lowercase-to-uppercase</td>
</tr>
<tr>
<td>4th</td>
<td>uppercase-to-lowercase</td>
</tr>
<tr>
<td>5th</td>
<td>character classification</td>
</tr>
<tr>
<td>6th</td>
<td>scanner translation</td>
</tr>
</tbody>
</table>

**Interaction:**

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment Control: Language Control

PROC OPTIONS

GROUP= LANGUAGECONTROL

**Valid in:**

configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:**

Environment Control: Language Control

**PROC OPTIONS GROUP=**

LANGUAGECONTROL

**Interaction:**

The TRANTAB= system option specifies a translation table to use for the SAS session, including file transfers. The TRANTAB statement specifies a customized translation table (for example, to map an EBCDIC character to an ASCII character) to apply to the character set in the SAS file that is being exported or transferred.
<table>
<thead>
<tr>
<th>Position</th>
<th>Type of Translation Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>7th</td>
<td>delta characters</td>
</tr>
<tr>
<td>8th</td>
<td>scanner character classification</td>
</tr>
<tr>
<td>9th</td>
<td>not used</td>
</tr>
<tr>
<td>10th</td>
<td>DBCS user table</td>
</tr>
</tbody>
</table>

**CAUTION:**

Do not change a translation table unless you are familiar with its purpose. Translation tables are used internally by the SAS supervisor to implement NLS. If you are unfamiliar with the purpose of translation tables, do not change the specifications without proper technical advice.

To change one table, specify null entries for the other tables. For example, to change the lowercase-to-uppercase table, which is third in the list, specify uppercase as follows:

```sas
options trantab = ( , , new-uppercase-table);
```

The other tables remain unchanged. The output from the OPTIONS procedure reflects the last specification for the TRANTAB= option and not the composite specification. Here is an example:

```sas
options trantab = ( , , new-uppercase-table);
options trantab = ( , , new-lowercase-table);
```

PROC OPTIONS shows that the value for TRANTAB= is

```
( , , new-lowercase-table),
```

but both the new-uppercase and new-lowercase tables are in effect.

**See Also**

Chapter 19, “TRANTAB Procedure,” on page 669

---

**VALIDMEMNAME= System Option**

Specifies the rules for naming SAS data sets, SAS data views, and item stores.

- **Valid in:** Configuration file, SAS invocation, OPTIONS statement, SAS System Options window
- **Category:** Files: SAS Files
- **PROC OPTIONS GROUP:** SASFILES
- **Default:** The shipped default is COMPATIBLE.
- **Applies to:** Base SAS engine and SPD Engine
- **Restrictions:** The VALIDMEMNAME= option is not supported by the tape engines V9TAPE, V8TAPE, V7TAPE, and V6TAPE. Regardless of the value of VALIDMEMNAME, a member name cannot end in the special character # followed by three digits. This is because it would conflict with the
naming conventions for generation data sets. Using such a member name results in an error.

**Note:** This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

---

**Syntax**

`VALIDMEMNAME=COMPATIBLE | EXTEND`

**Syntax Description**

**COMPATIBLE**

specifies that a SAS data set name, a SAS data view name, or an item store name must follow these rules:

- The length of the names can be up to 32 characters.
- Names must begin with a letter of the Latin alphabet (A–Z, a–z) or an underscore. Subsequent characters can be letters of the Latin alphabet, numerals, or underscores.
- Names cannot contain blanks or special characters except for the underscore.
- Names can contain mixed-case letters. SAS internally converts the member name to uppercase. Therefore, you cannot use the same member name with a different combination of uppercase and lowercase letters to represent different variables. For example, `customer`, `Customer`, and `CUSTOMER` all represent the same member name. How the name is saved on disk is determined by the operating environment.

**Alias**   COMPAT

**EXTEND**

specifies that a SAS data set name, a SAS data view name, or an item store name must follow these rules:

- Names can include national characters.
- The name can include special characters, except for the `/ \ * ? " < > | : - .` characters.

  **Note:** The SPD Engine does not allow ‘.’ (the period) anywhere in the member name.

- The name must contain at least one character (letters, numbers, valid special characters, and national characters).
- The length of the name can be up to 32 bytes.
- Null bytes are not allowed.
- Names cannot begin with a blank or a ‘.’ (the period).

  **Note:** The SPD Engine does not allow ‘$’ as the first character of the member name.

- Leading and trailing blanks are deleted when the member is created.
- Names can contain mixed-case letters. SAS internally converts the member name to uppercase. Therefore, you cannot use the same member name with a different combination of uppercase and lowercase letters to represent different variables. For example, `customer`, `Customer`, and `CUSTOMER` all represent the same
member name. How the name appears is determined by the operating environment.

**Requirement**

When VALIDMEMNAME=EXTEND, SAS data set names, SAS data view names, and item store names must be written as a SAS name literal if the name includes blank spaces, special characters, or national characters. If you use either the percent sign (%) or the ampersand (&), then you must use single quotation marks in the name literal in order to avoid interaction with the SAS Macro Facility. For more information, see “SAS Name Literals” in *SAS Language Reference: Concepts*.

**Operating environments**

For Windows and UNIX operating environments, all Base SAS windows support the extended rules when VALIDMEMNAME=EXTEND is set.

For Windows and UNIX operating environments, when you reference a SAS file directly by its physical name, the final embedded period is an extension delimiter. If a physical file reference includes a SAS member name that contains a period, you must add the file extension. For example, if you reference the data set name my.member as a physical file, you would add the file extension sas7bdat to the reference, as shown in this SET statement: `set './saslib/my.member.sas7bdat'`.

**z/OS specifics**

The windowing environment for Base SAS supports the extended rules in the Editor, Log, and Output windows when VALIDMEMNAME=EXTEND is set. Other SAS windows, such as the VIEWTABLE window, do not support the extended rules.

When you reference a SAS file directly by its physical name, the final embedded period is considered to be an extension delimiter only if what follows the period is a valid SAS extension. Otherwise, the period is considered to be part of the member name. For example, in the name my.member, member is considered part of the member name and not a file extension. In the name 'my.member.sas7bdat', the member name is 'my.member' and the file extension is sas7bdat.

**Tip**

The name is displayed in uppercase letters.

**See**

“How Many Characters Can I Use When I Measure SAS Name Lengths in Bytes?” in *SAS Language Reference: Concepts*

**Examples**

```sas
data "August Purchases"n;
```

```sas
data 'Años de empleo'n.;
```

**CAUTION**

Throughout SAS, using the name literal syntax with SAS member names that exceed the 32-byte limit or that have excessive embedded quotation marks might cause unexpected results. The intent of the VALIDMEMNAME=EXTEND system option is to enable compatibility with other DBMS member naming conventions, such as allowing embedded blanks and national characters.
Details

When VALIDMEMNAME= EXTEND, valid characters that are allowed in a SAS data set name, SAS data view name, and an item store name are extended to these characters:

- international characters
- characters supported by third-party databases
- characters that are commonly used in a filename

Only the DATA, VIEW, and ITEMSTOR SAS member types support the extension of characters. The other member types, such as CATALOG and PROGRAM, do not support the extended characters. INDEX and AUDIT types that exist only with the associated DATA member support extended characters.

See Also

- “Rules for Words and Names in the SAS Language” in SAS Language Reference: Concepts

System Options:

- “VALIDVARNAME= System Option” on page 610

VALIDVARNAME= System Option

Specifies the rules for valid SAS variable names that can be created and processed during a SAS session.

Valid in: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: SAS Files

PROC OPTIONS
GROUP= SASFILES

Default: The shipped default is V7.

Note: This option can be restricted by a site administrator. For more information, see “Restricted Options” in SAS System Options: Reference.

Syntax

VALIDVARNAME=V7 | UPCASE | ANY

Syntax Description

V7

specifies that variable names must follow these rules:

- The length of a SAS variable names can be up to 32 characters.
- The first character must begin with a letter of the Latin alphabet (A - Z, a - z) or the underscore. Subsequent characters can be letters of the Latin alphabet, numerals, or underscores.
- Trailing blanks are ignored. The variable name alignment is left-justified.
A variable name cannot contain blanks or special characters except for the underscore.

A variable name can contain mixed-case letters. SAS stores and writes the variable name in the same case that is used in the first reference to the variable. However, when SAS processes a variable name, SAS internally converts it to uppercase. Therefore, you cannot use the same variable name with a different combination of uppercase and lowercase letters to represent different variables. For example, `cat`, `Cat`, and `CAT` all represent the same variable.

Do not assign variables the names of special SAS automatic variables (such as `_N_` and `_ERROR_`) or variable list names (such as `_NUMERIC_`, `_CHARACTER_`, and `_ALL_`) to variables.

Examples

```sas
season='summer';
percent_of_profit=percent;
```

**UPCASE**

specifies that the variable name follows the same rules as V7, except that the variable name is uppercase, as in earlier versions of SAS.

**ANY**

specifies that SAS variable names must follow these rules:

- The name can begin with or contain any characters, including blanks, national characters, special characters, and multi-byte characters.
- The name can be up to 32 bytes in length
- The name cannot contain any null bytes
- Leading blanks are preserved, but trailing blanks are ignored
- The name must contain at least one character. A name with all blanks is not permitted.
- The name contain mixed-case letters. SAS stores and writes the variable name in the same case that is used in the first reference to the variable. However, when SAS processes a variable name, SAS internally converts it to uppercase. Therefore, you cannot use the same variable name with a different combination of uppercase and lowercase letters to represent different variables. For example, `cat`, `Cat`, and `CAT` all represent the same variable.

Requirement

If you use any characters other than the ones that are valid when the `VALIDVARNAME` system option is set to V7 (letters of the Latin alphabet, numerals, or underscores), then you must express the variable name as a name literal and you must set `VALIDVARNAME=ANY`. If the name includes either the percent sign (%) or the ampersand (&), then you must use single quotation marks in the name literal in order to avoid interaction with the SAS Macro Facility. See “SAS Name Literals” in *SAS Language Reference: Concepts* and “Avoiding Errors When Using Name Literals” in *SAS Language Reference: Concepts*.

See

“How Many Characters Can I Use When I Measure SAS Name Lengths in Bytes?” in *SAS Language Reference: Concepts*

Examples

```sas
\(\%\) of profit’n=percent;
items@warehouse’n=itemnum;
```
CAUTION Throughout SAS, using the name literal syntax with SAS member names that exceed the 32-byte limit or have excessive embedded quotation marks might cause unexpected results. The intent of the VALIDVARNAME=ANY system option is to enable compatibility with other DBMS variable (column) naming conventions, such as allowing embedded blanks and national characters.

See Also

- “Rules for Words and Names in the SAS Language” in SAS Language Reference: Concepts

System Options:

- “VALIDMEMNAME= System Option” on page 607

URLENCODING= System Option

Specifies whether the argument to the URLENCODE function and to the URLDECODE function is interpreted using the SAS session encoding or UTF-8 encoding.

**Valid in:** Configuration file, SAS invocation, OPTIONS statement, SAS System Options window

**Category:** Environment Control: Language Control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**See:** “URLENCODING= System Option” in SAS System Options: Reference

**Syntax**

URLENCODING=SESSION | UTF8
Part 9

Options for Commands, Statements, and Procedures for NLS

Chapter 16

Dictionary of Command, Statement, and Procedure Option for NLS
Chapter 16

Dictionary of Command, Statement, and Procedure Option for NLS

Commands, Statements, and Procedures for NLS by Category

The data set control and data access categories of options for selected SAS statements are affected by NLS. The following table provides brief descriptions of the statement options. For more detailed descriptions, see the dictionary entry for each statement option:

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Access</td>
<td>CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options (p. 625)</td>
<td>Specifies attributes for character variables that are needed in order to transcode a SAS file.</td>
</tr>
<tr>
<td></td>
<td>ENCODING= Option (p. 630)</td>
<td>Overrides and transcodes the encoding for input or output processing of external files.</td>
</tr>
<tr>
<td></td>
<td>INENCODING= and OUTENCODING= Options (p. 634)</td>
<td>Overrides and changes the encoding when reading or writing SAS data sets in the SAS library.</td>
</tr>
</tbody>
</table>
### Dictionary

#### CHARSET= Option

Specifies the character set to be generated in the META declaration for the output.

- **Valid in:** LIBNAME statement for the ODS MARKUP and ODS HTML statements
- **Category:** ODS: Third-Party Formatted

**Syntax**

```
CHARSET=character-set;
```

**Required Argument**

- **character-set**
  
  Specifies the character set to use in the META tag for HTML output.

An example of an encoding is ISO-8859-1. Official character sets for use on the Internet are registered by IANA (Internet Assigned Numbers Authority). IANA is the central registry for various Internet protocol parameters, such as port, protocol and enterprise numbers, and options, codes and types. For a complete list of character-set values, see [www.unicode.org/reports/tr22/index.html](http://www.unicode.org/reports/tr22/index.html) and [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets).
A character set is like an encoding-value in this context. However, character set is the term that is used to identify an encoding that is suitable for use on the Internet.

**Example: Generated Output in a META Declaration for an ODS MARKUP Statement**

```html
<META http-equiv="Content-Type" content="text/html; charset=iso-8858-1">
```

**See Also**

**Conceptual Information:**
- Chapter 3, “Encoding for NLS,” on page 9

**Statements:**

---

### Collating Sequence Option

Specifies the collating sequence for PROC SORT.

**Valid in:** PROC SORT statement

**Note:** The PROC SORT statement sorts observations in a SAS data set by one or more characters or numeric variables.

---

### Syntax

```sas
PROC SORT collating-sequence-option <other option(s)> ;
```

**Options**

Options can include one `collating-sequence-option` and multiple `other option(s)`. The order of the two types of options does not matter and both types are not necessary in the same PROC SORT step. Only the explanations for the PROC SORT collating-sequence-options follow.

**Operating Environment Information**

For information about behavior specific to your operating environment for the DANISH, FINNISH, NORWEGIAN, or SWEDISH `collating-sequence-option`, see the SAS documentation for your operating environment.

#### ASCII

sorts character variables using the ASCII collating sequence. You need this option only when you want to achieve an ASCII ordering on a system where EBCDIC is the native collating sequence.

#### DANISH NORWEGIAN

sorts characters according to the Danish and Norwegian

The Danish and Norwegian collating sequence is shown in Figure 16.1 on page 619.
**EBCDIC**
sorts character variables using the EBCDIC collating sequence. You need this option only when you want to achieve an EBCDIC ordering on a system where ASCII is the native collating sequence.

**POLISH**
sorts characters according to the Polish convention.

**FINNISH SWEDISH**
sorts characters according to the Finnish and Swedish convention. The Finnish and Swedish collating sequence is shown in Figure 16.1 on page 619.

**NATIONAL**
sorts character variables using an alternate collating sequence, as defined by your installation, to reflect a country's National Use Differences. To use this option, your site must have a customized national sort sequence. Check with the SAS Installation Representative at your site to determine whether a customized national sort sequence is available.

**NORWEGIAN**
See DANISH

**SWEDISH**
See FINNISH

**SORTSEQ=collating-sequence**
specifies the collating sequence. The *collating-sequence* can be a collating-sequence-option, a translation table, an encoding, or the keyword LINGUISTIC. Only one collating sequence can be specified. For more information, see “Collating Sequence” on page 17.

Here are descriptions of the collating sequences:

```
collating—sequence—option | translation_table
```
specifies either a translation table, which can be one that SAS provides or any user-defined translation table, or one of the PROC SORT statement Collating-Sequenc-Sequence-Options. For an example of using PROC TRANTAB and PROC SORT with SORTSEQ=, see “Example 6: Using Different Translation Tables for Sorting” on page 688.

The available translation tables are

- ASCII
- DANISH
- EBCDIC
- FINNISH
- ITALIAN
- NORWEGIAN
- POLISH
- REVERSE
- SPANISH
- SWEDISH

The following figure shows how the alphanumeric characters in each language sorts:
Restriction  You can specify only one collating-sequence-option in a PROC SORT step.

Tip  The SORTSEQ= collating sequence options are specified without parenthesis and have no arguments that are associated with them. An example of how to specify a collating sequence follows:  

```plaintext
proc sort data=mydata SORTSEQ=ASCII;
```

```
encoding-value
```

specifies an encoding value. The result is the same as a binary collation of the character data represented in the specified encoding. See the supported encoding values in “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

Restriction  PROC SORT is the only procedure or part of the SAS system that recognizes an encoding specified for the SORTSEQ= option.

Tip  When the encoding value contains a character other than an alphanumeric character or underscore, the value needs to be enclosed in quotation marks.

See  The list of the encodings that can be specified in “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

```
LINGUISTIC<collating—rules>
```

specifies linguistic collation, which sorts characters according to rules of the specified language. The rules and default collating sequence options are based on the language specified in the current locale setting. The implementation is provided by the International Components for Unicode (ICU) library and produces results that are largely compatible with the Unicode Collation Algorithms (UCA).

Alias  UCA

Restriction  The SORTSEQ=LINGUISTIC option is available only on the PROC SORT SORTSEQ= option and is not available for the SAS System SORTSEQ= option.

Note  Starting in the third maintenance release of SAS 9.4, linguistic collation can also be specified using the SORTSEQ= option in the SQL Procedure and by specifying system option SORTSEQ=LINGUISTIC. For more information, see “SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 602 and “SORT” in Base SAS Procedures Guide.
Linguistic sorting requires more memory with the z/OS mainframe. You might need to set your REGION to 50M or higher. This action must be done in JCL, if you are running in batch mode, or in the VERIFY screen if you are running interactively. This action allows the ICU libraries to load properly and does not affect the memory that is used for sorting.

The collating-rules must be enclosed in parentheses. More than one collating rule can be specified.

When BY processing is performed on data sets that are sorted with linguistic collation, the NOBYSORTED system option might need to be specified in order for the data set to be treated properly. BY processing is performed differently than collating sequence processing.

See

The “ICU License” in SAS Companion for z/OS

The “Collating Sequence” on page 17 for detailed information about linguistic collation.

“SORTSEQ=sort-table | LINGUISTIC” in SAS SQL Procedure User's Guide for information about linguistic sorting in PROC SORT.


The following are the collation-rules that can be specified for the LINGUISTIC option. These rules modify the linguistic collating sequence:

**ALTERNATE_HANDLING=SHIFTED**

controls the handling of variable characters like spaces, punctuation, and symbols. When this option is not specified (using the default value NON_IGNORABLE), differences among these variable characters are of the same importance as differences among letters. If the ALTERNATE_HANDLING option is specified, these variable characters are of minor importance.

**Default** NON_IGNORABLE

**Tip** The SHIFTED value is often used in combination with STRENGTH= set to Quaternary. In such a case, whitespace characters, punctuation, and symbols are considered when comparing strings, but only if all other aspects of the strings (base letters, accents, and case) are identical.

**CASE_FIRST=**

specify order of uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL levels. The following table provides the values and information for the CASE_FIRST argument:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER</td>
<td>Sorts uppercase letters first, then the lowercase letters.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOWER</td>
<td>Sorts lowercase letters first, then the uppercase letters.</td>
</tr>
</tbody>
</table>

**COLLATION=**

The following table lists the available COLLATION= values: If you do not select a collation value, then the user's locale-default collation is selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIG5HAN</td>
<td>specifies pinyin ordering for Latin and specifies big5 charset ordering for Chinese, Japanese, and Korean characters.</td>
</tr>
<tr>
<td>DIRECT</td>
<td>specifies a Hindi variant.</td>
</tr>
<tr>
<td>GB2312HAN</td>
<td>specifies pinyin ordering for Latin and specifies gb2312han charset ordering for Chinese, Japanese, and Korean characters.</td>
</tr>
<tr>
<td>PHONEBOOK</td>
<td>specifies a telephone-book style for ordering of characters. Select PHONEBOOK only with the German language.</td>
</tr>
<tr>
<td>PINYIN</td>
<td>specifies an ordering for Chinese, Japanese, and Korean characters based on character-by-character transliteration into pinyin. This ordering is typically used with simplified Chinese.</td>
</tr>
<tr>
<td>POSIX</td>
<td>is the Portable Operating System Interface. This option specifies a &quot;C&quot; locale ordering of characters.</td>
</tr>
<tr>
<td>STROKE</td>
<td>specifies a nonalphabetic writing style ordering of characters. Select STROKE with Chinese, Japanese, Korean, or Vietnamese languages. This ordering is typically used with Traditional Chinese.</td>
</tr>
<tr>
<td>TRADITIONAL</td>
<td>specifies a traditional style for ordering of characters. For example, select TRADITIONAL with the Spanish language.</td>
</tr>
</tbody>
</table>

**LOCALE=locale_name**

specifies the locale name in the form of a POSIX name. For example, ja_JP. See the “LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options” on page 697 for a list of locale and POSIX values supported by PROC SORT.

**Restriction**

The following locales are not supported by PROC SORT:

- Afrikaans_SouthAfrica, af_ZA
**NUMERIC_COLLATION**

orders integer values within the text by the numeric value instead of characters used to represent the numbers.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Order numbers by the numeric value. For example, &quot;8 Main St.&quot; would sort before &quot;45 Main St.&quot;.</td>
</tr>
<tr>
<td>OFF</td>
<td>Order numbers by the character value. For example, &quot;45 Main St.&quot; would sort before &quot;8 Main St.&quot;.</td>
</tr>
</tbody>
</table>

Default: OFF

**STRENGTH**

The value of strength is related to the collation level. There are five collation-level values. The following table provides information about the five levels. The default value for strength is related to the locale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Type of Collation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY</td>
<td>PRIMARY specifies differences between base characters (for example, &quot;a&quot; &lt; &quot;b&quot;).</td>
<td>It is the strongest difference. For example, dictionaries are divided into different sections by base character.</td>
</tr>
<tr>
<td>or 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECONDARY</td>
<td>Accents in the characters are considered secondary differences (for example, &quot;âs&quot; &lt; &quot;âs&quot; &lt; &quot;ât&quot;).</td>
<td>A secondary difference is ignored when there is a primary difference anywhere in the strings. Other differences between letters can also be considered secondary differences, depending on the language.</td>
</tr>
<tr>
<td>or 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TERTIARY</td>
<td>Upper and lowercase differences in characters are distinguished at the tertiary level (for example, &quot;ao&quot; &lt; &quot;Ao&quot; &lt; &quot;ao&quot;).</td>
<td>A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings. Another example is the difference between large and small Kana.</td>
</tr>
<tr>
<td>or 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUATERNARY</td>
<td>When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, &quot;âb&quot; &lt; &quot;a-b&quot; &lt; &quot;âB&quot;).</td>
<td>The quaternary level should be used if ignoring punctuation is required or when processing Japanese text. This difference is ignored when there is a primary, secondary, or tertiary difference.</td>
</tr>
<tr>
<td>or 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>Type of Collation</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IDENTICAL or 5</td>
<td>When all other levels are equal, the identical level is used as a tiebreaker. The Unicode code point values of the Normalization Form D (NFD) form of each string are compared at this level, just in case there is no difference at levels 1-4.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This level should be used sparingly, as only code point values differences between two strings is an extremely rare occurrence. For example, only Hebrew cantillation marks are distinguished at this level.</td>
<td></td>
</tr>
</tbody>
</table>

Alias Level=

**CAUTION:**

If you use a host sort utility to sort your data, then specifying a translation table based collating sequence with the SORTSEQ= option might corrupt the character BY variables. For more information, see the PROC SORT documentation for your operating environment.

**Details**

The collating sequence option in the PROC SORT statement sorts observations in a SAS data set by one or more characters or numeric variables.

**Table 16.1 Options**

<table>
<thead>
<tr>
<th>Task</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the collating sequence</td>
<td></td>
</tr>
<tr>
<td>Specify ASCII</td>
<td>ASCII on page 617</td>
</tr>
<tr>
<td>Specify EBCDIC</td>
<td>EBCDIC on page 618</td>
</tr>
<tr>
<td>Specify Danish</td>
<td>DANISH on page 617</td>
</tr>
<tr>
<td>Specify Finnish</td>
<td>FINNISH on page 618</td>
</tr>
<tr>
<td>Specify Norwegian</td>
<td>NORWEGIAN on page 617</td>
</tr>
<tr>
<td>Specify Polish</td>
<td>POLISH on page 618</td>
</tr>
<tr>
<td>Specify Swedish</td>
<td>SWEDISH on page 618</td>
</tr>
<tr>
<td>Specify a customized sequence</td>
<td>NATIONAL on page 618</td>
</tr>
</tbody>
</table>
Task | Option
--- | ---
Specify any of the collating sequences listed above (ASCII, EBCDIC, DANISH, FINNISH, ITALIAN, NORWEGIAN, POLISH, SPANISH, SWEDISH, or NATIONAL), the name of any other system provided translation table (POLISH, SPANISH), and the name of a user-created translation table. You can specify an encoding. You can also specify either the keyword LINGUISTIC or UCA to achieve a locale-appropriate collating sequence. | SORTSEQ= on page 618

See Also
- “Collating Sequence” on page 17
- “SORT” in *Base SAS Procedures Guide*

System Options:
- “SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 602
- “TRANTAB= System Option” on page 605

**CORRECTENCODING= Option**

Explicitly changes the encoding attribute of a SAS file to match the encoding of the data in the SAS file.

**Valid in:** MODIFY statement of the DATASETS procedure

**Syntax**

```sql
MODIFY SAS file </<CORRECTENCODING=encoding-value> > ;
```

**Optional Argument**

`</<CORRECTENCODING=encoding-value> >`

enables you to change the encoding indicator, which is recorded in the file's descriptor information, in order to match the actual encoding of the file's data. You cannot use this option in parenthesis after the name of each SAS file; you must specify CORRECTENCODING= after the forward slash. For example:

```sql
modify mydata / correctencoding=latin2;
```

For a list of valid encoding values for transcoding, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

**Restriction**

CORRECTENCODING= can be used only when the SAS file uses the default base engine, which is V9 in SAS 9.
Example: Using the CORRECTENCODING= Option to Resolve a SAS Session Encoding and a SAS File Encoding

A file's encoding indicator can be different from the data's encoding. For example, a SAS file that was created before SAS 9 has no encoding indicator stored on the file. If such a SAS file that has no recorded encoding is opened in a SAS 9 session, SAS assigns the encoding of the current session. For example, if the encoding of the data is Danish EBCDIC, but the encoding for the current session is Western Wlatin1, then the actual encoding of the file's data and the encoding indicator that is stored in the file's descriptor information do not match. When this action occurs, the data does not transcode correctly and could result in unreadable output. The following MODIFY statement would resolve the problem by explicitly assigning an EDCDIC encoding:

Note: CEDA creates a read-only copy. You need to copy the data with PROC COPY or a DATA step to transcode the data permanently.

```
proc datasets library=myfiles;
  modify olddata / correctencoding=ebcdic1142;
quit;
```

CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options

Specifies attributes for character variables that are needed in order to transcode a SAS file.

Valid in: LIBNAME statement
Category: Data Access
PROC OPTIONS GROUP= LIBNAME statement under Windows, UNIX, and Z/OS in the documentation for your operating environment.
See: LIBNAME, SAS/ACCESS

Syntax

```
LIBNAME libref <CVPBYTES=bytes> <CVPENGINE=engine> <CVPMULTIPLIER=multiplier> 'SAS data-library';
```

Optional Arguments

```
CVPBYTES=bytes
```

specifies the number of bytes by which to expand character variable lengths when processing a SAS data file that requires transcoding. The CVP engine expands the lengths so that character data truncation does not occur. The lengths for character variables are increased by adding the specified value to the current length. You can specify a value from 0 to 32766.

For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPBYTES= option.

```
libname expand 'SAS data-library' cvpbytes=5;
```

Character variable lengths are increased by adding 5 bytes. A character variable with a length of 10 is increased to 15, and a character variable with a length of 100 is increased to 105.
If you specify CVPBYTES=, SAS automatically uses the CVP engine in order to expand the character variable lengths according to your specification. If you explicitly assign the CVP engine but do not specify either CVPBYTES= or CVPMULTIPLIER=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths of the character variables.

The CVP engine supports SAS data files, no SAS views, catalogs, item stores, and so on.

The CVP engine is available for input (read) processing only.

For library concatenation with mixed engines that include the CVP engine, only SAS data files are processed. For example, if you execute the COPY procedure, only SAS data files are copied.

The number of bytes that you specify must be large enough to accommodate any expansion. Otherwise, truncation occurs, which results in an error message in the SAS log.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

The number of bytes that you specify must be large enough to accommodate any expansion. Otherwise, truncation occurs, which results in an error message in the SAS log.

You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

“The Avoiding Character Data Truncation By Using the CVP Engine” on page 34

specifies the engine to use in order to process a SAS data file that requires transcoding. The CVP engine expands the character variable lengths to transcoding so that character data truncation does not occur. Then the specified engine does the actual file processing.

SAS uses the default SAS engine.

“The Avoiding Character Data Truncation By Using the CVP Engine” on page 34

specifies a multiplier value in order to expand character variable lengths when you are processing a SAS data file that requires transcoding. The CVP engine expands the lengths so that character data truncation does not occur. The lengths for character variables are increased by multiplying the current length by the specified value. You can specify a multiplier value from 1 to 5.

For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPMULTIPLIER= option.

```
libname expand 'SAS data-library' cvpmultiplier=2.5;
```

Character variable lengths are increased by multiplying the lengths by 2.5. A character variable with a length of 10 is increased to 25, and a character variable with a length of 100 is increased to 250.

If you specify CVPMULTIPLIER=, SAS automatically uses the CVP engine in order to expand the character variable lengths according to
your specification. If you explicitly specify the CVP engine but do not specify either CVPMULTIPLIER= or CVPPBYTES=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths.

**Restrictions**
The CVP engine supports SAS data files, no SAS views, catalogs, item stores, and so on.

The CVP engine is available for input (read) processing only.

For library concatenation with mixed engines that include the CVP engine, only SAS data files are processed. For example, if you execute the COPY procedure, only SAS data files are copied.

**Requirement**
The number of bytes that you specify must be large enough to accommodate any expansion. Otherwise, truncation occurs, which results in an error in the SAS log.

**Interaction**
You cannot specify both CVPMULTIPLIER= and CVPPBYTES=. Specify one of these options.

**See**
“Avoiding Character Data Truncation By Using the CVP Engine” on page 34

### Example: Using the CVP (Character Variable Padding) Engine

The following example illustrates how to avoid character data truncation by using the CVP engine. The example uses a SAS data set named MYFILES.WLATIN2, which contains some national characters in Wlatin2 encoding.

```sas
libname myfiles 'C:\Documents and Settings\sasdxw\My Documents\myfiles';
data myfiles.wlatin2 (encoding=wlatin2);
  var1='41'x;
  var2='8a'x;
  var3='9c'x;
  var4='b3'x;
;
proc print data=myfiles.wlatin2;
run;
```

<table>
<thead>
<tr>
<th>The SAS System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Here is PROC CONTENTS output for MYFILES.WLATIN2, which shows that the encoding is WLatin2 and that the length for each character variable is 1 byte:
Output 16.1  PROC CONTENTS Output for MYFILES.WLATIN2

The following code is executed with the session encoding Wlatin2.

options msglevel=i;
libname myfiles 'SAS data-library';
data myfiles.utf8 (encoding="utf-8");
   set myfiles.wlatin2;
run;

The DATA step requests a new data set named MYFILES.UTF8, and requests that the
data be read into the new data set in UTF-8 encoding, which means that the data must be
transcoded from Wlatin2 to UTF-8. The request results in errors due to character data
truncation that occurs from the transcoding. The new data set MYFILES.UTF8 is
created but does not contain any data.
Log 16.1  SAS Log with Transcoding Error

```
1   options msglevel=i;
2   libname myfiles 'C:\Documents and Settings\xxxxxx\My Documents\myfiles';
   NOTE: Libref MYFILES was successfully assigned as follows:
      Engine:        V9
      Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
3   data myfiles.utf8 (encoding="utf-8");
4      set myfiles.wlatin2;
5    run;
INFO: Data file MYFILES.UTF8.DATA is in a format native to another host or the file encoding does not match the session encoding.
Cross Environment Data Access will be used, which may require additional CPU resources and reduce performance.
ERROR: Some character data was lost during transcoding in the data set MYFILES.UTF8.
   NOTE: The data step has been abnormally terminated.
   NOTE: The SAS System stopped processing this step because of errors.
   NOTE: There were 1 observations read from the data set MYFILES.WLATIN2.
   WARNING: The data set MYFILES.UTF8 may be incomplete. When this step was stopped there were 0 observations and 4 variables.
```

The following code is executed again with the session encoding Wlatin2.

```
options msglevel=i;
libname myfiles 'SAS data-library';
libname expand cvp 'SAS data-library' cvpbytes=2;
data myfiles.utf8 (encoding="utf-8");
   set expand.wlatin2;
run;
```

In this example, the CVP engine is used to expand character variable lengths by adding two bytes to each length. The data is read into the new file in UTF-8 encoding by transcoding from Wlatin2 to UTF-8. There is no data truncation due to the expanded character variable lengths, and the new data set is successfully created:

Log 16.2  SAS Log Output for MYFILES.UTF8

```
12  options msglevel=i;
13  libname myfiles 'C:\Documents and Settings\xxxxxx\My Documents\myfiles';
   NOTE: Directory for library MYFILES contains files of mixed engine types.
   NOTE: Libref MYFILES was successfully assigned as follows:
      Engine:        V9
      Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
14  libname expand cvp 'C:\Documents and Settings\xxxxxx\My Documents\myfiles' cvpbytes=2;
   WARNING: Libname EXPAND refers to the same physical library as MYFILES.
   NOTE: Libref EXPAND was successfully assigned as follows:
      Engine:        CVP
      Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
15  data myfiles.utf8 (encoding="utf-8");
16      set expand.wlatin2;
17    run;
INFO: Data file MYFILES.UTF8.DATA is in a format native to another host or the file encoding does not match the session encoding.
Cross Environment Data Access will be used, which may require additional CPU resources and reduce performance.
   NOTE: There were 1 observations read from the data set EXPAND.WLATIN2.
   NOTE: The data set MYFILES.UTF8 has 1 observations and 4 variables.
```

Finally, here is PROC CONTENTS output for MYFILES.UTF8 showing that it is in UTF-8 encoding and that the length of each character variable is 3:
# ENCODING= Option

Overrides and transcodes the encoding for input or output processing of external files.

**Valid in:**  
%INCLUDE statement; FILE statement; FILENAME statement; FILENAME statement, EMAIL (SMTP) Access Method; INFILE statement; ODS statements; FILE command; INCLUDE command

**Category:** Data Access

## Syntax

`ENCODING= 'encoding-value'`
Optional Argument

ENCODING= 'encoding-value'
specifies the encoding to use for reading, writing, copying, or saving an external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read, write, copy, or save data using an external file, SAS transcodes the data from the session encoding to the specified encoding.

For details, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

Default SAS uses the current session encoding.

Details

The following table provides information about how the ENCODING option is used with the corresponding statement:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%INCLUDE statement:</td>
<td>reads SAS statements and data lines from the specified source file (not supported under z/OS).</td>
</tr>
<tr>
<td>FILE statement:</td>
<td>writes to an external file.</td>
</tr>
<tr>
<td>FILENAME statement:</td>
<td>reads from or writes to an external file.</td>
</tr>
<tr>
<td>FILENAME statement, EMAIL (SMTP) Access Method:</td>
<td>sends electronic mail programmatically.</td>
</tr>
<tr>
<td>INFILE statement:</td>
<td>reads from an external file.</td>
</tr>
<tr>
<td>ODS statements:</td>
<td>controls features of the Output Delivery System that are used to generate, store, or reproduce SAS procedure and DATA step output.</td>
</tr>
<tr>
<td>FILE command:</td>
<td>saves the contents of a window to an external file.</td>
</tr>
<tr>
<td>INCLUDE command:</td>
<td>Copies an external file into the current window.</td>
</tr>
</tbody>
</table>

Some encodings use a Byte-Order Mark (BOM). The BOM is generated when the encoding is specified. For the UTF-8 encoding, you must specify encoding=utf-8 on the filename and file DATA step statements in order for the BOM to be generated.
Examples

Example 1: Using the FILE Statement to Specify an Encoding for Writing to an External File

This example creates an external file from a SAS data set. The current session encoding is Wlatin1, but the external file's encoding needs to be UTF-8. By default, SAS writes the external file using the current session encoding.

To specify what encoding to use for writing data to the external file, specify the ENCODING= option:

```sas
libname myfiles 'SAS data-library';
filename outfile 'external-file';
data _null_
   set myfiles.cars;
   file outfile encoding="utf-8";
   put Make Model Year;
run;
```

When you tell SAS that the external file is to be in UTF-8 encoding, SAS then transcodes the data from Wlatin1 to the specified UTF-8 encoding.

Example 2: Using the FILENAME Statement to Specify an Encoding for Reading an External File

This example creates a SAS data set from an external file. The external file is in UTF-8 character-set encoding, and the current SAS session is in the Wlatin1 encoding. By default, SAS assumes that an external file is in the same encoding as the session encoding, which causes the character data to be written to the new SAS data set incorrectly.

To specify which encoding to use when reading the external file, specify the ENCODING= option:

```sas
libname myfiles 'SAS data-library';
filename extfile 'external-file' encoding="utf-8";
data myfiles.unicode;
   infile extfile;
   input Make $ Model $ Year;
run;
```

When you specify that the external file is in UTF-8, SAS then transcodes the external file from UTF-8 to the current session encoding when writing to the new SAS data set. Therefore, the data is written to the new data set correctly in Wlatin1.

Example 3: Using the FILENAME Statement to Specify an Encoding for Writing to an External File

This example creates an external file from a SAS data set. By default, SAS writes the external file using the current session encoding. The current session encoding is Wlatin1, but the external file's encoding needs to be UTF-8.

To specify which encoding to use when writing data to the external file, specify the ENCODING= option:

```sas
libname myfiles 'SAS data-library';
filename outfile 'external-file' encoding="utf-8";
data _null_
   set myfiles.cars;
```
file outfile;
  put Make Model Year;
run;

When you specify that the external file is to be in UTF-8 encoding, SAS then transcodes the data from WLatin1 to the specified UTF-8 encoding when writing to the external file.

Example 4: Changing Encoding for Message Body and Attachment
This example illustrates how to change text encoding for the message body as well as for the attachment.

```sas
filename mymail email 'Joe.Developer@sas.com';
data _null_;   
  file mymail
    subject='Text Encoding'
    encoding=greek
    attach=('C:\My Files\Test.out'
      content_type='text/plain'
      encoding='ebcdic1047'
      outencoding='latin1');
run;
```

In the program, the following occurs:

- The ENCODING= email option specifies that the message body is encoded to Greek (ISO) before being sent.
- For the ATTACH= email option, the attachment option ENCODING= specifies the encoding of the attachment that is read into SAS, which is Western (EBCDIC).
- Because SMTP and other email interfaces do not support EBCDIC, the attachment option OUTENCODING= converts the attachment to Western (ISO) before sending it.

Example 5: Using the INFILE= Statement to Specify an Encoding for Reading from an External File
This example creates a SAS data set from an external file. The external file's encoding is in UTF-8, and the current SAS session encoding is WLatin1. By default, SAS assumes that the external file is in the same encoding as the session encoding, which causes the character data to be written to the new SAS data set incorrectly.

To specify which encoding to use when reading the external file, specify the ENCODING= option:

```sas
libname myfiles 'SAS data-library';
filename extfile 'external-file';
data myfiles.unicode;
  infile extfile encoding="utf-8";
  input Make $ Model $ Year;
run;
```

When you specify that the external file is in UTF-8, SAS then transcodes the external file from UTF-8 to the current session encoding when writing to the new SAS data set. Therefore, the data is written to the new data set correctly in WLatin1.

See Also

Statements:
INENCODING= and OUTENCODING= Options

Overrides and changes the encoding when reading or writing SAS data sets in the SAS library.

**Valid in:** LIBNAME statement

**Category:** Data Access

### Syntax

**INENCODING=** ANY | ASCIIANY | EBCDICANY | encoding-value

**OUTENCODING=** ANY | ASCIIANY | EBCDICANY | encoding-value

### Syntax Description

**ANY**

specifies no transcoding between ASCII and EBCDIC encodings.

NOTE: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant.

**ASCIIANY**

specifies that no transcoding occurs, assuming that the mixed encodings are ASCII encodings.

**EBCDICANY**

specifies that no transcoding occurs, assuming that the mixed encodings are EBCDIC encodings.

**encoding-value**

specifies an encoding value. For a list of encoding values, see Chapter 23, “Encoding Values for a SAS Session,” on page 723.

### Details

The INENCODING= option is used to read SAS data sets in the SAS library. The OUTENCODING= option is used to write SAS data sets in the SAS library.
The INENCODING= value or the OUTENCODING= value is written to the SAS log when you use the LIST argument.

INENCODING= and OUTENCODING= are most appropriate when using an existing library that contains mixed encodings. To read a library that contains mixed encodings, you can set INENCODING= to ASCIANY or EBCDICANY. To write a separate data set, you can use OUTENCODING= to specify a specific encoding, which is applied to the data set when it is created.

Comparisons

- Session encoding is specified using the ENCODING= system option or the LOCALE= system option. Each operating environment has a default encoding.
- You can specify the encoding for reading data sets in a SAS library by using the LIBNAME statement INENCODING= option for input files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.
- You can specify the encoding for writing data sets to a SAS library by using the LIBNAME statement OUTENCODING= option for output files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.
- For the COPY procedure, the default CLONE option uses the encoding attribute of the input data set instead of the encoding value specified on the OUTENCODING= option. For more information about CLONE and NOCLONE, see COPY Statement.

Note: This interaction does not apply when using SAS/CONNECT or SAS/SHARE.

See Also

- “Overview: Encoding for NLS” on page 9

Statements:

- “LIBNAME Statement” in SAS Statements: Reference

System Options:

- “ENCODING System Option: UNIX, Windows, and z/OS” on page 587
- “LOCALE System Option” on page 591

Data Set Options:

- “ENCODING= Data Set Option” on page 57

ODSCHARSET= Option

Specifies the character set to be generated in the META declaration for the output.

Valid in: LIBNAME statement for the XML engine
Category: Data Access
Syntax
ODSCHARSET=character-set;

Required Argument
character-set
For the LIBNAME statement for the XML engine, specifies the character set to use in the ENCODING= attribute.

An example of an encoding is ISO-8859-1. Official character sets for use on the Internet are registered by IANA (Internet Assigned Numbers Authority). IANA is the central registry for various Internet protocol parameters, such as port, protocol and enterprise numbers, options, codes, and types. For a complete list of character-set values, see www.unicode.org/reports/tr22/index.html and www.iana.org/assignments/character-sets.

A character set is like an encoding-value in this context. However, character set is the term that is used to identify an encoding that is suitable for use on the Internet.

Details
An XML declaration is not required in all XML documents. Such a declaration is required only when the character encoding of the document is other than the default UTF-8 or UTF-16 and no encoding was determined by a higher-level protocol.

The ODSCHARSET option, in the LIBNAME statement for the XML engine, specifies the character set to use for generating an output XML document.

See Also
Conceptual Information:
• Chapter 3, “Encoding for NLS,” on page 9

Statements:
• SAS XML LIBNAME Engine: User's Guide

ODSTRANTAB= Option
Specifies the translation table to use when transcoding an XML document for an output file.

Valid in: the LIBNAME statement for the XML engine
Category: Data Access

Syntax
TRANTAB ='translation-table'

Optional Argument
translation-table
specifies the translation table to use for the output file. The translation table is an encoding method that maps characters (letters, logograms, digits, punctuation,
symbols, control characters, and so on) in the character set to numeric values. An example of a translation table is one that converts characters from EBCDIC to ASCII-ISO. The table-name can be any translation table that SAS provides, or any user-defined translation table. The value must be the name of a SAS catalog entry in either the SASUSER.PROFILE catalog or the SASHELP.HOST catalog.

Details

For SAS 9.2, using the ODSTRANTAB= option in the LIBNAME statement for the XML Engine is supported for backward compatibility. The preferred method for specifying an encoding is to use the LOCALE= system option.

See Also

Conceptual Information:
- “Transcoding and Translation Tables” on page 28
- Conceptual discussion of Chapter 2, “Locale for NLS,” on page 5

System Options:
- “TRANTAB= System Option” on page 605
- “LOCALE System Option” on page 591

Procedures:
- Chapter 19, “TRANTAB Procedure,” on page 669

Statements:

---

**TRANSCODE= Column Modifier on PROC SQL**

Specifies whether values can be transcoded for character columns.

**Valid in:** Column modifier component in the SQL Procedure

**Syntax**

TRANSCODE=YES|NO

**Required Argument**

TRANSCODE=YES|NO

for character columns, specifies whether values can be transcoded. Use TRANSCODE=NO to suppress transcoding. Note that when you create a table using the CREATE TABLE AS statement, the transcoding attribute for a particular character column in the created table is the same as it is in the source table unless you change it with the TRANSCODE= column modifier.

**Default**

YES
Restriction  Suppression of transcoding is not supported for the V6TAPE engine.

See Also  
• Chapter 4, “Transcoding for NLS,” on page 27  
• Base SAS Procedures Guide

RENCODING= Option  
Specifies the ASCII-based or EBCDIC-based encoding to use for transcoding data for a SAS/SHARE server session that is using an EBCDICANY or ASCIIANY session encoding.

Valid in: LIBNAME statement for SAS/SHARE only  
Category: Data Access  
Note: The RENCODING= option in the LIBNAME statement is relevant only if using a SAS/SHARE server that has a session encoding set to EBCDICANY or ASCIIANY to preserve a mixed-encoding computing environment.  
See: LIBNAME statement in SAS/SHARE User's Guide

Syntax  
RENCODING=ASCII-encoding-value | EBCDIC-encoding-value

Syntax Description  

ASCII-encoding-value  
For a list of valid values for ASCII encodings for UNIX and Windows, see Chapter 23, “Encoding Values for a SAS Session,” on page 723.

EBCDIC-encoding-value  
For a list of valid values for EBCDIC encodings for z/OS, see Chapter 23, “Encoding Values for a SAS Session,” on page 723.

Details  
If you use SAS/SHARE in a mixed-encoding environment (for example, SAS/SHARE client sessions using incompatible encodings such as Latin1 and Latin2), you can set the following options:

• in the SAS/SHARE server session, set the SAS system option ENCODING=EBCDICANY or ENCODING=ASCIIANY  
• in the SAS/SHARE client session, set the RENCODING= option in the LIBNAME statement(s) under these conditions:  
  • a client session that uses an ASCII-based encoding accesses an EBCDICANY server  
  • a client session that uses an EBCDIC-based encoding accesses an ASCIIANY server.

The RENCODING= option enables SAS/SHARE clients to specify which encoding to assume the server's data is in when transcoding to or from the client session encoding.
For SAS 9, if you are processing data in a SAS/SHARE client/server session from more than one SBCS or DBCS encoding, you are advised to use the UTF8 encoding. For more information about Unicode servers that run the UTF8 session encoding, go to http://rnd.sas.com/sites/i18n/i18ndocs/i18nsupport/Pages/SAS%20Technical%20Papers.aspx and search for Processing Multilingual Data with the SAS® 9.2 Unicode Server and Multilingual Computing with SAS® 9.4.

Comparisons

In SAS 9, you can maintain multilingual data that contains characters from more than one traditional SBCS or DBCS encoding in a SAS data set by using a UTF8 encoding. To share Update access to that data using SAS/SHARE, you must also run the SAS/SHARE server using a session encoding of UTF8. SAS transcodes the data to the client encoding if necessary.

Prior to SAS 9, if a SAS/SHARE client and a SAS/SHARE server ran on common architectures (for example, the client and server ran on UNIX machines), there was no automatic transcoding of character data. It was possible to build applications that accessed data sets in different EBCDIC or ASCII encodings within a single SAS/SHARE server, or that accessed data sets in mixed different encodings within a single data set. This method was very uncommon and required careful programming to set up transcoding tables from clients that ran in different operating environments.

The following steps describe how you can maintain mixed encoding in SAS 9, if necessary.

- The SAS/SHARE server must run by using a session encoding of EBCDICANY for mixed-EBCDIC encodings or ASCIIANY for mixed-ASCII encodings.

  This restores the behavior of Version 8 and earlier releases and prevent the automatic character transcoding between different client and server encodings in the same EBCDIC or ASCII family. That is, no transcoding occurs under these circumstances:

  - if the client session encoding is an EBCDIC encoding and the server session encoding is EBCDICANY
  - if the client session encoding is an ASCII encoding and the server session encoding is ASCIIANY.

- A SAS/SHARE client that does not share the same encoding family as an ASCIIANY or EBCDICANY server can control the necessary transcoding by using an RENCODING= option on the first LIBNAME statement that accesses the server.

  For example, an ASCII client that runs in a Polish locale could access a z/OS EBCDICANY server and specify RENCODING=EBCDIC870 to access data that the client knows contains Polish-encoded data. Another ASCII client that runs in a German locale could access the same z/OS EBCDICANY server and specify RENCODING=EBCDIC1141 to access data that the client knows contains German data. Similarly, EBCDIC clients that access an ASCIIANY server can specify the precise ASCII encoding of the data that they are accessing by using the RENCODING= option in the LIBNAME statement.

See Also

Conceptual Information:

- “Overview to Transcoding” on page 27
**TRANSCODE= Option**

Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and length with one or more variables) that indicates whether character variables are to be transcoded.

**Valid in:** the ATTRIB statement in a DATA step  
**Category:** Information  
**Type:** Declarative  
**See:** ATTRIB Statement under Windows UNIX z/OS in the documentation for your operating environment.

**Syntax**

\[ \text{ATTRIB } \text{variable-list(s) attribute-list(s) ;} \]

**Required Arguments**

- **variable-list**
  names the variables that you want to associate with the attributes.  
  **Tip** List the variables in any form that SAS allows.

- **attribute-list**
  specifies one or more attributes to assign to variable-list. Multiple attributes can be specified in the ATTRIB statement. For a complete list of attributes, see the “ATTRIB Statement” in *SAS Statements: Reference* .

**TRANSCODE= YES | NO**

Specifies whether to transcode character variables. Use TRANSCODE=NO to suppress transcoding. For more information, see “Overview to Transcoding” on page 27.

**Default** YES

**Restriction** The TRANSCODE=NO attribute is not supported by some SAS Workspace Server clients. Variables with TRANSCODE=NO are not returned in SAS 9.4. Prior to SAS 9.4, variables with TRANSCODE=NO are transcoded. Prior releases of SAS cannot access a SAS 9.4 data set that contains a variable with a TRANSCODE=NO attribute.

**Interactions** You can use the VTRANSCODE and VTRANSCODEX functions to return whether transcoding is on or off for a character variable.

If the TRANSCODE= attribute is set to NO for any character variable in a data set, PROC CONTENTS prints a transcoded column that contains the TRANSCODE= value for each variable in the data set. If all variables in the data set are set to the default TRANSCODE= value (YES), no transcoded column is printed.
Examples

Example 1: Using the TRANSCODE= Option with the SET Statement
When you use the SET statement to create a data set from several data sets, SAS makes the TRANSCODE= attribute of the variable in the output data set equal to the TRANSCODE= value of the variable in the first data set. In this example, the variable Z's TRANSCODE= attribute in data set A is NO because B is the first data set and Z's TRANSCODE= attribute in data set B is NO.

```plaintext
data b;
  length z $4;
  z = 'ice';
  attrib z transcode = NO;

data c;
  length z $4;
  z = 'snow';
  attrib z transcode = YES;

data a;
  set b;
  set c;
  /* Check transcode setting for variable Z */
  rc1 = vtranscode(z);
  put rc1=;
run;
```

Example 2: Using the TRANSCODE= Option with the MERGE Statement
When you use the MERGE statement to create a data set from several data sets, SAS makes the TRANSCODE= attribute of the variable in the output data set equal to the TRANSCODE= value of the variable in the first data set. In this example, the variable Z's TRANSCODE= attribute in data set A is YES because C is the first data set and Z's TRANSCODE= attribute in data set C is YES.

```plaintext
data b;
  length z $4;
  z = 'ice';
  attrib z transcode = NO;

data c;
  length z $4;
  z = 'snow';
  attrib z transcode = YES;

data a;
  merge c b;
  /* Check transcode setting for variable Z */
  rc1 = vtranscode(z);
  put rc1=;
run;
```

Note: The TRANSCODE= attribute is set when the variable is first seen on an input data set or in an ATTRIB TRANSCODE= statement. If a SET or MERGE statement comes before an ATTRIB TRANSCODE= statement and the TRANSCODE= attribute contradicts the SET statement, an error message occurs.
TRANTAB= Option

Specifies the translation table to use when you are transcoding character data.

Valid in: ODS MARKUP statement and ODS RTF statement
Category: ODS: Third-Party Formatted

Syntax

TRANTAB = (translation-table)

Optional Argument

translation-table

specifies the translation table to use for the output file. The translation table is an encoding method that maps characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) in the character set to numeric values. An example of a translation table is one that converts characters from EBCDIC to ASCII-ISO. The table-name can be any translation table that SAS provides, or any user-defined translation table. The value must be the name of a SAS catalog entry in either the SASUSER.PROFILE catalog or the SASHELP.HOST catalog.

Details

In SAS 9.4, the TRANTAB= option in the ODS MARKUP statement is supported for backward compatibility. The ENCODING= option is preferred when specifying the encoding.

See Also

Conceptual Information:

• “Transcoding and Translation Tables” on page 28

Procedures:

• Chapter 19, “TRANTAB Procedure,” on page 669

Statements:

**XMENCODING= Option**

Overrides the encoding of an XML document to import or export an external document.

Valid in: LIBNAME statement for the XML engine

Category: Data Access

**Syntax**

```
XMENCODING= 'encoding-value'
```

**Details**

The LIBNAME statement for the XML engine, associates a SAS libref with an XML document to import or export an external document.

**Comparisons**

Options

`encoding-value`

specifies the encoding to use when you read, write, copy, or save an external file.

The value for XMENCODING= indicates that the external file has a different encoding from the current session encoding.

For details, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 713.

The default for `encoding-value` is the current session encoding.

**See Also**

**Statements:**


---

**TRANTAB Statement**

Specifies the translation table to use when you transcode character data in order to export or transfer a SAS file.

Valid in: CPOR Procedure, UPLOAD procedure, DOWNLOAD procedure

Restriction: You can specify only one translation table per TRANTAB statement. To specify additional translation tables, use additional TRANTAB statements.

Interaction: The TRANTAB statement specifies a customized translation table (for example, to map an EBCDIC character to an ASCII character) to apply to the character set in the SAS file that is being exported or transferred. The TRANTAB= system option specifies a translation table to use for the SAS session, including file transfers.
Syntax

TRANTAB NAME=translation-table-name <TYPE=(etype-list) <OPT=DISP | SRC | (DISP SRC)> > ;

Required Argument

NAME=translation-table-name

specifies the name of the translation table to apply to the SAS catalog that you want to export (PROC CPORT) or transfer (PROC UPLOAD or PROC DOWNLOAD). The translation-table-name that you specify as the name of a catalog entry in either your SASUSER.PROFILE catalog or the SASHELP.HOST catalog. The SASUSER.PROFILE catalog is searched first, and then the SASHELP.HOST catalog is searched.

In most cases, the default translation table is the correct one to use, but you might need to apply additional translation tables if, for example, your application requires different national language characters.

You can specify a translation table other than the default in two ways:

• To specify a translation table for an invocation of the procedure, use the TRANTAB statement in the procedure, as appropriate.

• To specify a translation table for your entire SAS session or job (including all file exports or transfers), use the TRANTAB= system option.

Optional Arguments

TYPE=(etype-list)

applies the translation table only to the entries with the type or types that you specify. The etype-list can be one or more entry types. Examples of catalog entry types include DATA and FORMAT. If etype-list is a simple entry type, omit the parentheses.

By default, the UPLOAD, DOWNLOAD, and CPORT procedures apply the translation table to all specified catalog entries.

OPT=DISP | SRC | (DISP SRC)

OPT=DISP applies the translation table only to the specified catalog entries, which produce window displays.

OPT=SRC applies the translation table only to the specified catalog entries that are of the type SOURCE.

OPT=(DISP SRC) applies the translation table only to the specified catalog entries that either produce window displays or are of type SOURCE.

If you do not specify the OPT= option, the UPLOAD or DOWNLOAD procedure applies the translation table to all of the entries in the catalog that you specify.

Default PROC CPORT, PROC UPLOAD, and PROC DOWNLOAD apply the translation table to all entries and data sets in the specified catalog.

Details

Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on direct use of translation tables. SAS 9.3 supports the TRANTAB statement for backward
compatibility. However, using the LOCALE= system option is preferred in later SAS releases. For more information, see TS-639, Data Conversion Issues in V6–V8. This technical support note provides information for customers using non-English languages.


PROC CPORT is used when you transfer a SAS file across a network. PROC UPLOAD and PROC DOWNLOAD are used when you transfer a SAS file across a network.

You must specify the INCAT= and OUTCAT= options in the PROC UPLOAD or PROC DOWNLOAD statement when using the TRANTAB statement.

Examples

Example 1

The information that follows applies to procedure features:

- PROC CPORT statement option: FILE=
- TRANTAB statement option: TYPE=

This example shows how to apply a customized translation table to the transport file before PROC CPORT exports it. For this example, assume that you have already created a customized translation table called TTABLE1.

Example 2: Program

Assign library references. The LIBNAME andFILENAME statements assign a libref for the source library and a fileref for the transport file, respectively.

```
libname source '\\sashq\root\pub\pubdoc\doc\901\authoring\proc\miscsrc\sasfiles\cport';
filename tranfile 'trans3';
proc trantab table=ascii;
save table=ttable1;
```

Apply the translation specifics. The TRANTAB statement applies the translation that you specify with the customized translation table TTABLE1. TYPE= limits the translation to FORMAT entries.

```
proc cport catalog=source.formats file=tranfile;
   trantab name=ttable1 type=(format);
run;
```

Example 3: SAS Log

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FORMATS
NOTE: The catalog has 2 entries and its maximum logical record length is 104.
NOTE: Entry REVENUE.FORMAT has been transported.
NOTE: Entry DEPT.FORMATC has been transported.
```
See Also

Conceptual Information:
• Chapter 4, “Transcoding for NLS,” on page 27

System Options:
• “TRANTAB= System Option” on page 605

Procedures:
• Chapter 19, “TRANTAB Procedure,” on page 669
• “CPORT” in Base SAS Procedures Guide
• “UPLOAD” in SAS/CONNECT User's Guide
• “DOWNLOAD” in SAS/CONNECT User's Guide
Part 10

Procedures for NLS

Chapter 17
DBCSTAB Procedure .................................................. 649

Chapter 18
LOCALEDATA Procedure ........................................... 655

Chapter 19
TRANTAB Procedure .................................................. 669
Overview: DBCSTAB Procedure

The DBCSTAB procedure produces conversion tables for the double-byte character sets (DBCSs) that SAS supports.

Use the DBCSTAB procedure to modify an existing DBCS table when

- the DBCS encoding system that you are using is not supported by SAS
- the DBCS encoding system that you are using has a nonstandard translation table

You might be more likely to use the DBCSTAB procedure when a valid DBCSTYPE= value is not available. These values depend on the operating environment. In such cases, you can use the DBCSTAB procedure to modify a similar translation table, and then you can specify the use of the new table with the TRANTAB option.

Syntax: DBCSTAB Procedure

PROC DBCSTAB TABLE=table-name
  <BASETYPE=base-type> <CATALOG=<libref>:catalog-name> <DATA=<libref>:table-name> <DBCSLANG=language>
  <DESC='description'> <FORCE> <VERIFY> <VERBOSE>; 

PROC DBCSTAB Statement

Produces conversion tables for double-byte character sets.
Syntax

PROC DBCSTAB TABLE=table-name
<option(s)>;

Required Argument

TABLE=table-name
specifies the name of the double-byte code table to produce. This table name becomes an entry of type DBCSTAB in the catalog that is specified with the CATALOG= option. By default, the catalog name is Sasuser.DBCS.

Alias  NAME=, N=

Optional Arguments

BASETYPE=base-type
specifies a base type for the double-byte code table conversion. If you use this option, you reduce the number of tables that are produced.

If you specify BASETYPE=, then all double-byte codes are first converted to the base code, and then they are converted to the required code. If you have n codes, then there are n(n-1) conversions that must be made.

Alias  BTYPE=

CATALOG=<libref.>catalog-name
specifies the name of the catalog in which the table is stored. If the catalog does not exist, it is created.

Default  Sasuser.DBCS

DATA=<libref.>table-name
specifies the data for producing the double-byte code table. Several double-byte character variables are required to produce the table. Use variable names that are equivalent to the value of the DBCSTYPE= system option and recognized by the KCVT function.

DBCSLANG=language
specifies the language that the double-byte code table uses. The value of this option should match the value of the DBCSLANG= system option.

Alias  DBLANG=

DESC='description'
specifies a text string to put in the DESCRIPTION field for the entry.

FORCE
produces the conversion tables, even if errors are present.

VERIFY
checks the data range of the input table based on the code. This option checks for invalid double-byte code.

VERBOSE
prints the statistics detail when building DBCS tables.
The following example creates a Japanese translation table called CUSTAB and demonstrates how the TRANTAB= option specifies the new translation table. The DBCS=, DBCSLANG=, and DBCSTYPE= options are specified at start-up.

Program

```sas
proc dbcstab
    /* name of the new translate table */
    name=custtab
    /* based on pcibm encoding */
    basetype=pcms
    /* data to create the new table */
    data=trantab
    /* japanese language */
    dbcslang=japanese
    /* catalog descriptor */
    desc='Modified Japanese Trantab'
    /* where the table is stored */
    catalog=sasuser.dbcs
    /* checks for invalid DBCS in the new data */
    verify;
run;
```

Creating a Conversion Table

Invoke Japanese SAS with Shift_JIS encoding.

```sas
data trantab;
pcms='8342'x; dec='b9b3'x;
run;
proc dbcstab;
name=custtab;
basetype=pcms;
data=trantab;
dbcslang=japanese;
desc='Modified Japanese Trantab';
```
catalog=sasuser.dbcs;
verify;
run;
options trantab=({}, ,,, ,,, custtab);

Program Description

Set the SAS environment.
   invoke Japanese SAS with Shift_JIS encoding.

Create the TRANTAB data set.
   data trantab;

Define the pcms and dec parameters and run the code.
   pcms='8342'x; dec='b9b3'x;
   run;

Invoke the DBCSTAB procedure.
   proc dbcstab;

Assign a name to a translate table.
   name=custtab;

Apply pcibm encoding.
   basetype=pcms;

Access data from the TranTab data set.
   data=trantab;

Set the language to Japanese.
   dbcslang=japanese;

Set the catalog descriptor to Modified Japanese Translation table.
   desc='Modified Japanese Trantab';

Set the table's location.
   catalog=sasuser.dbcs;

Check for invalid DBCS data and then process the program.
   verify;
   run;

Select the translate table using the TRANTAB= option. Translate tables are used for DBCS conversion with SAS/CONNECT, PROC CPORT and PROC CIMPORT, and the DATA step function KCVT. The TRANTAB= option can specify DBCS translate tables. For SAS 8.2 and earlier, the ninth argument specified the DBCS system table. However, for SAS 9 and later, instead of using the ninth argument, SAS uses a system table that is
Example 2: Producing Japanese Conversion Tables with the DBCSTAB Procedure

Features: PROC DBCSTAB statement options:
- TABLE=
- DATA=
- DBLANG=
- BASETYPE=
- VERIFY

Program
```sas
data ja_jpn;
length ibm jis euc pcibm $2.;
ibm='4040'x;
jis='2121'x;
euc='a1a1'x;
pcibm='8140'x;
run;
proc dbcstab;
table=japanese;
data=ja_jpn;
dblang=japanese;
basetype=jis;
verify;
run;
```

Program Description

Define the ja_jpn data set.
```sas
data ja_jpn;
```

Set the length of the specified encodings to 2.
```sas
length ibm jis euc pcibm $2.;
```

Assign the value 4040 to the ibm encoding.
```sas
ibm='4040'x;
```

Assign the value 2121 to the jis encoding.
```sas
jis='2121'x;
```
Assign the value a1a1 to the euc encoding.
```
euc='a1a1'x;
```

Assign the value 8140 to the pcibm encoding and run the code.
```
pcibm='8140'x;
run;
```

Invoke the DBCSTAB procedure.
```
proc dbcstab;
```

Define the table with the name, Japanese.
```
table=japanese;
```

Access the Ja_Jpn data set.
```
data=ja_jpn;
```

Specify Japanese for the language.
```
dblang=japanese;
```

Specify the jis encoding.
```
basetype=jis;
```

Check for invalid DBCS data and then process the program.
```
verify;
run;
```

SAS Log
```
1   proc dbcstab
2   table=ja_jpn
3   data=work.ja_jpn
4   dblang=japanese
5   basetype=jis
6   verify;
7   run;

NOTE: Base table for JIS created.
NOTE: IBM table for JIS created.
NOTE: PCIBM table for JIS created.
NOTE: EUC table for JIS created.
NOTE: Base table for IBM created.
NOTE: JIS table for IBM created.
NOTE: Base table for PCIBM created.
NOTE: JIS table for PCIBM created.
NOTE: Base table for EUC created.
NOTE: JIS table for EUC created.
NOTE: 10 DBCS tables are generated. Each table has 1 DBCS characters.
NOTE: Each table is 2 bytes in size.
NOTE: Required table memory size is 612.
NOTE: There were 1 observations read from the data set WORK.JA_JPN.
```
Chapter 18
LOCALEDATA Procedure

Overview: LOCALEDATA Procedure
The LOCALEDATA procedure lets you customize data. You can view, open, modify, and store the customized locale data.

Syntax: LOCALEDATA Procedure
PROC LOCALEDATA;
    LOAD SASLOCALE | REGISTRY;
    MODIFY key=key-name value=key-value | category=category-name value=locale;
    CONTENTS _ALL_ | key-name | category-name;
    SAVE REGISTRY < _ALL_ | key-name | category-name syntax=SAS >;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC LOCALEDATA</td>
<td>Specifies the source of customized locale data.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Specifies how to customize the locale data by locale element key or by locale category.</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>Specifies what locale element’s values are discarded.</td>
</tr>
<tr>
<td>SAVE</td>
<td>Stores the customized locale data in the SAS registry.</td>
</tr>
</tbody>
</table>
### PROC LOCALEDATA Statement

Specifies the source of customized locale data.

**Syntax**

```
PROC LOCALEDATA;
```

### LOAD Statement

Specifies which source of locale data is loaded for customization.

**Default:** Registry

**Syntax**

```
LOAD SASLOCALE | REGISTRY;
```

**Required Arguments**

- **SASLOCALE**
  - loads the locale data from the factory setting default locale database.
- **REGISTRY**
  - loads the locale data from the SAS registry. This locale data could be customized locale data that has been stored in the SAS registry.

### MODIFY Statement

Customizes the locale data by locale element key or by locale category.

**Syntax**

```
MODIFY key=key-name value=key-value | category=category-name value=locale
```

**Required Arguments**

- **key=key-name** **value=key-value**
  - customizes single locale element values.
- **category=category-name** **value=locale**
  - customizes all locale elements in a category. You can select one of the following categories:
The following table shows the element keys that you can modify.
The value of the locale element key must be less than the maximum length value.
The following values can be used for the type:

0     string
1     unassigned integer; use double quotation marks

<table>
<thead>
<tr>
<th>Locale Element Key</th>
<th>Max Length</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATESTYLE</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PAPERSIZE</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTITLE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FTEXT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMFONT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SORTSEQ</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>MESSAGES</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_DATEETE</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_TIME</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_NUMERIC</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORMATNAME_PERCENT</td>
<td>512</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_SANS_SERIF</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_Cursive</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_Fantasy</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FONT_MONOSPACE</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>BRUSH</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SIMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COMPLEX</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SWISS</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ITALIC</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DATE_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME AMPM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME_WEEK_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATETIME WEEK_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME AMPM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>TIME_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE WEEK_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE WEEK_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE YYMM_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE YYMM_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE MMDD_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE MMDD_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>DATE_YEAR_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YEAR_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYQQ_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_YYWW_SHORT_FORMAT</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DATE_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABMON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON01</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON02</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON03</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON04</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>MON05</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON06</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON07</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON08</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON09</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON10</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON11</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>MON12</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABDAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY5</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY6</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>DAY7</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>AM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>PM</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>Locale Element Key</td>
<td>Max Length</td>
<td>Type</td>
<td>Category</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>ABQTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>ABQTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR1</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR2</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR3</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>QTR4</td>
<td>512</td>
<td>0</td>
<td>LC_TIME</td>
</tr>
<tr>
<td>INT_CURRENCY_SYMBOL</td>
<td>3</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>CURRENCY_SYMBOL</td>
<td>32</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_DECIMAL_POINT</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_THOUSANDS_SEP</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_GROUPING</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_POSITIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_NEGATIVE_SIGN</td>
<td>8</td>
<td>0</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_INT_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_FRAC_DIGITS</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_P_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_CS_PRECEDES</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
<tr>
<td>MON_N_SEP_BY_SPACE</td>
<td>3</td>
<td>1</td>
<td>LC_MONETARY</td>
</tr>
</tbody>
</table>
### CONTENTS Statement

Displays the value of the specified locale element key.

#### Syntax

```
CONTENTS _ALL_ | locale-element-key | category-name
```

#### Required Arguments

- **_ALL_**
  - displays the value of all locale element keys.

- **locale-element-key**
  - displays the value of the specified locale element key.

- **category-name**
  - displays the value of locale elements for the specified category.
SAVE Statement
Stores the customized locale data in the SAS registry.

Syntax
SAVE REGISTRY < _ALL_ | key-name | category-name syntax=SAS>

Required Argument
REGISTRY
stores locale data in the Sasuser library and registers it in the SAS registry. The encoding of the stored locale data is in UNICODE escape. Here is the hierarchy of the saved locale element in the SAS registry: SAS_REGISTRY ⇒ LOCALE ⇒ sas locale ⇒ locale element key locale category ⇒ locale element key.

Optional Arguments
_ALL_
saves all locale data elements.
key-name
saves the specified locale element key name and value.
category-name syntax=SAS
saves all elements in the specified category.

Example: Modifying Locale Data Using the LOCALEDATA Procedure

Features:
PROC LOCALEDATA statement
LOAD statement
MODIFY statement
SAVE statement
CONTENTS statement

Other features:
DATA step
PUT statement

The following program modifies locale data. SAS is invoked twice.

Modifying Locale Definitions

PROC LOCALEDATA;
LOAD SASLOCALE;
MODIFY KEY=QTR1 VALUE='QTR1';
MODIFY KEY=QTR2 VALUE='QTR2';
MODIFY KEY=QTR3 VALUE='QTR3';
MODIFY KEY=QTR4 VALUE='QTR4';
MODIFY KEY=DATE_SHORT_FORMAT VALUE='%Y/%m/%d';
MODIFY KEY=DATE_YEAR_FORMAT VALUE='Year(%Y)';
SAVE REGISTRY / _ALL_ syntax=SAS;
CONTENTS _ALL_;
QUIT;
data _null_
  a = 19208;
  put a nldate. / a nldateyq. / a nldateyr.;
run;

Program Description

Invoke the LOCALEDATA procedure.

PROC LOCALEDATA;

Load the locale definitions from the local locale database.

LOAD SASLOCALE;

Modify the definition values.

MODIFY KEY=QTR1 VALUE='QTR1';
MODIFY KEY=QTR2 VALUE='QTR2';
MODIFY KEY=QTR3 VALUE='QTR3';
MODIFY KEY=QTR4 VALUE='QTR4';
MODIFY KEY=DATE_SHORT_FORMAT VALUE='%Y/%m/%d';
MODIFY KEY=DATE_YEAR_FORMAT VALUE='Year(%Y)';

Save the modified definitions in the registry. This action overwrites the definitions that are in the registry so that there are no duplicate key values.

SAVE REGISTRY / _ALL_ syntax=SAS;

View the modified locale definitions from the registry.

CONTENTS _ALL_;

Invoke the QUIT command to end processing.

QUIT;

Create a DATA step for testing the modified locale definitions.

data _null_

Assign a value of 19208 to variable a.

  a = 19208;

Output the value of variable a using the formats NLDATE., NLDATEYQ., and NLDATEYR.

  put a nldate. / a nldateyq. / a nldateyr.;
run;
Example: Modifying Locale Data Using the LOCALEDATA Procedure

SAS Log

1    PROC LOCALEDATA;
2
3      /* Load locale definition from locale DB */
4      LOAD SASLOCALE;
5
6      /* Change the value of the elements */
7      MODIFY KEY=QTR1 VALUE='QTR1';
8      MODIFY KEY=QTR2 VALUE='QTR2';
9      MODIFY KEY=QTR3 VALUE='QTR3';
10     MODIFY KEY=QTR4 VALUE='QTR4';
11     MODIFY KEY=DATE_SHORT_FORMAT VALUE='YY//m/d';
12     MODIFY KEY=DATE_YEAR_FORMAT VALUE='Year(YY)';
13     /* Store the locale definition into registry, this will override the
14       existing definition,
15       so there is no duplicate key issue */
16     SAVE REGISTRY / _ALL_ syntax=SAS;
17
18     /* View the locale definition from registry */
19     CONTENTS _ALL_;
20     LOCALE="English_UnitedStates"
21     LANGUAGE="English"
22     LANGUAGE_SCRIPT="en"
23     TERRITORY="UnitedStates"
24     LANGID="en"
25     LOCID="US"
26     DATESTYLE="MDY"
27     PAPERSIZE="LETTER"
28     TITLE=""
29     FTEXT=""
30     SIMFONT=""
31     SORTSEQ=""
32     MESSAGES=""
33     FORMATNAME_DATE="NLDATE16.
34     FORMATNAME_DATETIME="NLDATM24.
35     FORMATNAME_TIME="NLTIME8.
36     FORMATNAME_NUMERIC="BEST12.
37     FORMATNAME_PERCENT="PERCENT12.
38     FONT_SERIF="COMPLEX"
39     FONT_SANS Serif="SIMPLEX"
40     FONT_CURSIVE="ITALIC"
41     FONT_FANTASY="BRUSH"
42     FONT_MONOSPACE="SIMPLEX"
43     BRUSH="BRUSH"
44     SIMPLEX="SIMPLEX"
45     COMPLEX="COMPLEX"
46     SWISS="SWISS"
47     ITALIC="ITALIC"
48     DATE_FORMAT="%B %d, %Y"
49     DATE_SHORT_FORMAT="%Y/%m/%d"
50     DATE_MEDIUM_FORMAT="%b %e, %Y"
51     DATE_LONG_FORMAT="%B %e, %Y"
52     DATE_FULL_FORMAT="%A, %B %e, %Y"
53     DATE_FULL_AB_FORMAT="%a, %b %e, %Y"
54     DATE_YYQQ_FORMAT="%Q %Y"
55     DATE_YYQQ_SHORT_FORMAT="%q %Y"
56     DATE_YYWW_FORMAT="Week %U %Y"
57     DATE_YYWW_SHORT_FORMAT="W%U %y"
Example: Modifying Locale Data Using the LOCALEDATA Procedure

```
ABDAY1="Sun"
ABDAY2="Mon"
ABDAY3="Tue"
ABDAY4="Wed"
ABDAY5="Thu"
ABDAY6="Fri"
ABDAY7="Sat"
DAY1="Sunday"
DAY2="Monday"
DAY3="Tuesday"
DAY4="Wednesday"
DAY5="Thursday"
DAY6="Friday"
DAY7="Saturday"
SA_ABDAY1="Sun"
SA_ABDAY2="Mon"
SA_ABDAY3="Tue"
SA_ABDAY4="Wed"
SA_ABDAY5="Thu"
SA_ABDAY6="Fri"
SA_ABDAY7="Sat"
SA_DAY1="#Sunday"
SA_DAY2="#Monday"
SA_DAY3="#Tuesday"
SA_DAY4="#Wednesday"
SA_DAY5="#Thursday"
SA_DAY6="#Friday"
SA_DAY7="#Saturday"
ABQTR1="#Q1"
ABQTR2="#Q2"
ABQTR3="#Q3"
ABQTR4="#Q4"
QTR1="#QTR1"
QTR2="#QTR2"
QTR3="#QTR3"
QTR4="#QTR4"
SA_ABQTR1="#Q1"
SA_ABQTR2="#Q2"
SA_ABQTR3="#Q3"
SA_ABQTR4="#Q4"
SA_QTR1="#1st quarter"
SA_QTR2="#2nd quarter"
SA_QTR3="#3rd quarter"
SA_QTR4="#4th quarter"
AM="#AM"
PM="#PM"
DATE_SEP="/" 
FIRST_DAY_OF_WEEK="#0"
INT_CURRENCY_SYMBOL="USD"
CURRENCY_SYMBOL="#$
MON_DECIMAL_POINT="#.
MON_THOUSANDS_SEP="#,
MON_GROUPING="#3"
MON_POSITIVE_SIGN="#"
MON_NEGATIVE_SIGN="#-
MON_INT_FRAC_DIGITS="#2"
MON_FRAC_DIGITS="#2"
MON_P_CS_PRECEDES="#1"
MON_P_SEP_BY_SPACE="#0"
MON_N_CS_PRECEDES="#1"
MON_N_SEP_BY_SPACE="#0"
MON_P_SIGN_POSN="#1"
MON_N_SIGN_POSN="#0"
```
Chapter 18 • LOCALEDATA Procedure

20
21 data _null_;  
22 a = 19208;  
23 put a nldate. / a NLDATEYQ. / a NLDATEYR.;  
24 run ;  

August 03, 2012
3rd quarter 2012
2012

NOTE: DATA statement used (Total process time):
   real time           2.96 seconds  
   cpu time            0.28 seconds  

18
19 QUIT;  

NOTE: PROCEDURE LOCALEDATA used (Total process time):
   real time           0.24 seconds  
   cpu time            0.18 seconds  

18
19 QUIT;
Overview: TRANTAB Procedure

The TRANTAB procedure creates, edits, and displays customized translation tables. In addition, you can use PROC TRANTAB to view and modify translation tables that are supplied by SAS. These SAS supplied tables are stored in the Sashelp.Host catalog. Any translation table that you create or customize is stored in your Sasuser.Profile catalog. Translation tables have an entry type of TRANTAB.

Translation tables are operating environment-specific SAS catalog entries that translate the values of one (coded) character set to another. A translation table has two halves: table 1 provides a translation, such as ASCII to EBCDIC; table 2 provides the inverse (or reverse) translation, such as EBCDIC to ASCII. Each half of a translation table is an array of 256 two-digit positions, each of which contains a one-byte unsigned number that corresponds to a coded character.
SAS uses translation tables to
- determine the collating sequence in the SORT procedure
- facilitate data communications between the operating environment and a graphics device when you run SAS/GRAPH software in an IBM environment
- accommodate national language character sets other than U.S. English.

PROC TRANTAB produces no output. It can display translation tables and notes in the SAS log.

Note: Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option to improve the direct use of translation tables. SAS 9.2 supports the TRANTAB procedure for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. PROC TRANTAB is an interactive procedure. Once you submit a PROC TRANTAB statement, you can continue to enter and execute statements without repeating the PROC TRANTAB statement. To terminate the procedure, submit a QUIT statement or another DATA or PROC statement.

Concepts: TRANTAB Procedure

Understanding Translation Tables and Character Sets for PROC TRANTAB

The Kth element in a translation table corresponds to the Kth element of an ordered character set. For example, position 00 (which is byte 1) in a translation table contains a coded value that corresponds to the first element of the ordered character set. To determine the position of a character in your operating environment's character set, use the RANK function. The following example shows how to use RANK:

```sas
data _null_
  x=rank('a');
  put "The position of a is " x ".";
run;
```

The SAS log prints the following message: The position of a is 97.

Each position in a translation table contains a hexadecimal number that is within the range of 0 ('00'x) to 255 ('FF'x). Hexadecimal values always end with an x. You can represent one or more consecutive hexadecimal values within quotation marks followed by a single x. For example, a string of three consecutive hexadecimal values can be written as '08090A'x. The SAS log displays each row of a translation table as 16 hexadecimal values enclosed in quotation marks followed by an x. The SAS log also lists reference numbers in the vertical and horizontal margins that correspond to the positions in the table. “Example 1: Viewing a Translation Table” on page 678 shows how the SAS log displays a translation table.

Storing Translation Tables with PROC TRANTAB

When you use PROC TRANTAB to create a customized translation table, the procedure automatically stores the table in your Sasuser.Profile catalog. This lets you use customized translation tables without affecting other users. When you specify the translation table in the SORT procedure or in a GOPTIONS statement, the software first
looks in your Sasuser.Profile catalog to find the table. If the specified translation table is not in your Sasuser.Profile catalog, the software looks in the Sashelp.Host catalog.

To access your translation table globally, your SAS Installation Representative must copy the table from your Sasuser.Profile catalog (using the CATALOG procedure) to the Sashelp.Host catalog. If the table is not found there, the software continues to search in Sashelp.Locale for it.

**Modifying SAS Translation Tables with PROC TRANTAB**

If a translation table that is provided by SAS does not meet your needs, you can use PROC TRANTAB to edit it and create a new table. That is, you can issue the PROC TRANTAB statement that specifies the SAS table, edit the table, and then save the table using the SAVE statement. The modified translation table is saved in your Sasuser.Profile catalog. If you are a SAS Installation Representative, you can modify a translation table with PROC TRANTAB and then use the CATALOG procedure to copy the modified table from your Sasuser.Profile catalog to the Sashelp.Host catalog, as shown in the following example:

```plaintext
proc catalog c=sasuser.profile;
   copy out=sashelp.host entrytype=trantab;
run;
```

You can use PROC TRANTAB to modify translation tables stored in the Sashelp.Host catalog only if you have update (or write) access to that data library and catalog.

**Using Translation Tables Outside PROC TRANTAB**

**Using Translation Tables in the SORT Procedure**

PROC SORT uses translation tables to determine the sort’s collating sequence. You can specify an alternative translation table with the SORTSEQ= option of PROC SORT. For example, if your operating environment sorts with the EBCDIC sequence by default, and you want to sort with the ASCII sequence, you can issue the following statement to specify the ASCII translation table:

```plaintext
proc sort sortseq=ascii;
run;
```

You can also create a customized translation table with PROC TRANTAB and specify the new table with PROC SORT. This table is useful when you want to specify sorting sequences for languages other than U.S. English.

See “Example 6: Using Different Translation Tables for Sorting” on page 688 for an example that uses translation tables to sort data in different ways. For information about the tables available for sorting and the SORTSEQ= option, see “SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 602.

**Transcoding External Files**

Translation tables are implicitly set by the LOCALE= system option. They are used only for transcoding external files.

You can list the translation tables with the OPTIONS procedure. The following example uses the z/OS environment with a LOCALE= system option set to en_US:

```plaintext
TRANTAB=(eol1wlt1,wlt1eol1,elat_ucs,elat_lcs,elat_ccl,,,elat_scc)
```
Using Translation Tables in SAS/GRAPH Software

In SAS/GRAPH software, translation tables are most commonly used in an IBM operating environment, where tables are necessary because graphics commands must leave IBM operating environments in EBCDIC representation but must reach asynchronous graphics devices in ASCII representation. Specifically, SAS/GRAPH software builds the command stream for these devices internally in ASCII representation. However, it must convert the commands to EBCDIC representation before they can be given to the communications software for transmission to the device. SAS/GRAPH software uses a translation table internally to make the initial conversion from ASCII to EBCDIC. The communications software then translates the command stream back to ASCII representation before it reaches the graphics device.

Translation tables are operating environment-specific. In most cases, you can simply use the default translation table, SASGTAB0, or one of the SAS supplied graphics translation tables. However, if these tables cannot perform all of the translation correctly, you can create your own translation table with PROC TRANTAB. The SASGTAB0 table might fail to do the translation correctly when it encounters characters from languages other than U.S. English.

To specify an alternative translation table for SAS/GRAPH software, you can either use the TRANTAB= option in a GOPTIONS statement or modify the TRANTAB device parameter in the device entry. For example, the following GOPTIONS statement specifies the GTABTCAM graphics translation table:

```
goptions trantab=gtabtcam;
```

Translation tables used in SAS/GRAPH software perform both device-to-operating environment translation and operating environment-to-device translation. Therefore, a translation table consists of 512 bytes, with the first 256 bytes used to perform device-to-operating environment translation (ASCII to EBCDIC on IBM mainframes) and the second 256 bytes used to perform operating environment-to-device translation (EBCDIC to ASCII on IBM mainframes). For PROC TRANTAB, the area of a translation table for device-to-operating environment translation is considered to be table 1, and the area for operating environment-to-device translation is considered to be table 2. See “Example 1: Viewing a Translation Table” on page 678 for a listing of the ASCII translation table (a SAS provided translation table), which shows both areas of the table.

For operating environments other than IBM mainframes, translation tables can be used to translate specific characters in the data stream that are created by the driver. For example, if the driver normally generates a vertical bar in the data stream, but you want another character to be generated in place of the vertical bar, you can create a translation table that translates the vertical bar to an alternate character.

SAS/GRAPH software also uses key maps and device maps to map codes generated by the keyboard to specified characters and to map character codes to codes required by the graphics output device. These maps are specific to SAS/GRAPH software. For more information, contact SAS Technical Support.

---

Syntax: TRANTAB Procedure

**Tip:** Supports RUN group processing
PROC TRANTAB TABLE=table-name <NLS>;
  CLEAR <ONE | TWO | BOTH>;
  INVERSE;
  LIST <ONE | TWO | BOTH>;
  LOAD TABLE=table-name <NLS>;
  REPLACE position value-1 < …value-n>;
  SAVE <TABLE=table-name> <CAT=libname.catalog > <ONE | TWO | BOTH>;
  SWAP;

Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option to improve direct use of translation tables. SAS 9.3 supports the TRANTAB procedure for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. PROC TRANTAB is an interactive procedure. Once you submit a PROC TRANTAB statement, you can continue to enter and execute statements without repeating the PROC TRANTAB statement. To terminate the procedure, submit a QUIT statement or another DATA or PROC statement.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC TRANTAB</td>
<td>Creates, edits, or displays a translations table</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Sets all positions in the translation table to 0</td>
</tr>
<tr>
<td>INVERSE</td>
<td>Creates an inverse of table 1</td>
</tr>
<tr>
<td>LIST</td>
<td>Displays a translation table in hexadecimal representation</td>
</tr>
<tr>
<td>LOAD</td>
<td>Loads a translation table into memory for editing</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Replaces the characters in a translation table with specified values</td>
</tr>
<tr>
<td>SAVE</td>
<td>Saves the translation table in your Sasuser.Profile catalog</td>
</tr>
<tr>
<td>SWAP</td>
<td>Exchanges table 1 with table 2</td>
</tr>
</tbody>
</table>

**PROC TRANTAB Statement**

Creates, edits, or displays a translation table.

**Tip:** If there is an incorrect table name in the PROC TRANTAB statement, use the LOAD statement to load the correct table. You do not need to reinvoke PROC TRANTAB. New tables are not stored in the catalog until you issue the SAVE statement, so you do not have unwanted tables in your catalog.

**Syntax**

PROC TRANTAB TABLE=table-name <NLS>;

**Syntax**

PROC TRANTAB TABLE=table-name <NLS>;

**Translation tables** were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option to improve direct use of translation tables. SAS 9.3 supports the TRANTAB procedure for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. PROC TRANTAB is an interactive procedure. Once you submit a PROC TRANTAB statement, you can continue to enter and execute statements without repeating the PROC TRANTAB statement. To terminate the procedure, submit a QUIT statement or another DATA or PROC statement.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC TRANTAB</td>
<td>Creates, edits, or displays a translations table</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Sets all positions in the translation table to 0</td>
</tr>
<tr>
<td>INVERSE</td>
<td>Creates an inverse of table 1</td>
</tr>
<tr>
<td>LIST</td>
<td>Displays a translation table in hexadecimal representation</td>
</tr>
<tr>
<td>LOAD</td>
<td>Loads a translation table into memory for editing</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Replaces the characters in a translation table with specified values</td>
</tr>
<tr>
<td>SAVE</td>
<td>Saves the translation table in your Sasuser.Profile catalog</td>
</tr>
<tr>
<td>SWAP</td>
<td>Exchanges table 1 with table 2</td>
</tr>
</tbody>
</table>

**PROC TRANTAB Statement**

Creates, edits, or displays a translation table.

**Tip:** If there is an incorrect table name in the PROC TRANTAB statement, use the LOAD statement to load the correct table. You do not need to reinvoke PROC TRANTAB. New tables are not stored in the catalog until you issue the SAVE statement, so you do not have unwanted tables in your catalog.

**Syntax**

PROC TRANTAB TABLE=table-name <NLS>;

**Translation tables** were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option to improve direct use of translation tables. SAS 9.3 supports the TRANTAB procedure for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. PROC TRANTAB is an interactive procedure. Once you submit a PROC TRANTAB statement, you can continue to enter and execute statements without repeating the PROC TRANTAB statement. To terminate the procedure, submit a QUIT statement or another DATA or PROC statement.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC TRANTAB</td>
<td>Creates, edits, or displays a translations table</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Sets all positions in the translation table to 0</td>
</tr>
<tr>
<td>INVERSE</td>
<td>Creates an inverse of table 1</td>
</tr>
<tr>
<td>LIST</td>
<td>Displays a translation table in hexadecimal representation</td>
</tr>
<tr>
<td>LOAD</td>
<td>Loads a translation table into memory for editing</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Replaces the characters in a translation table with specified values</td>
</tr>
<tr>
<td>SAVE</td>
<td>Saves the translation table in your Sasuser.Profile catalog</td>
</tr>
<tr>
<td>SWAP</td>
<td>Exchanges table 1 with table 2</td>
</tr>
</tbody>
</table>
**Required Argument**

**TABLE=**<table-name>

specifies the translation table to create, edit, or display the table. The specified table name must be a valid one-level SAS name with no more than eight characters.

**Optional Argument**

**NLS**

specifies that the table listed in the **TABLE=** argument is one of five special internal translation tables provided with SAS. You must use the NLS option when you specify one of the five special tables in the **TABLE=** argument. NLS stands for National Language Support. This option and the associated translation tables provide a method to translate characters that exist in languages other than U.S. English. To use the modified NLS table, specify its name in the SAS system option **TRANTAB=**. When you load one of these special translation tables, the SAS log displays a note stating that table 2 is uninitialized. That is, table 2 is an empty table that contains all 0s. PROC TRANTAB does not use table 2 for translation in these special cases.

- **SASXPT** specifies the local-to-transport format translation table (used by the CPORT procedure).
- **SASLCL** specifies the transport-to-local format translation table (used by the CIMPORT procedure).
- **SASUCS** specifies the lowercase-to-uppercase translation table (used by the UPCASE function).
- **SASLCS** specifies the uppercase-to-lowercase translation table (used by the LOWCASE macro).
- **SASCCL** specifies the character classification table (used internally), which contains flag bytes that correspond to each character position. These positions indicate the class or classes to which each character belongs.

---

**CLEAR Statement**

Sets all positions in the translation table to 0; used when you create a new table.

**Syntax**

```
CLEAR <ONE | TWO | BOTH>;
```

**Optional Argument**

**ONE | TWO | BOTH**

- **ONE** clears table 1.
TWO clears table 2.

BOTH clears both table 1 and table 2.

Default  ONE

---

**INVERSE Statement**

Creates an inverse of table 1 in a translation table. The INVERSE statement creates table 2.

**Syntax**

INVERSE;

**Details**

The INVERSE statement does not preserve multiple translations. Suppose table 1 has 2 (or more) different characters translated to the same value; for example, "A" and "B" are both translated to "1". For table 2, the INVERSE statement uses the last translated character for the value. "1" is always translated to "B" and not "A", assuming that "A" appears before "B" in the first table. Sort programs in SAS require an inverse table for proper operation.

---

**LIST Statement**

Displays a translation table in hexadecimal representation in the SAS log.

**Syntax**

LIST <ONE | TWO | BOTH>;

**Optional Argument**

ONE | TWO | BOTH

ONE  
displays table 1.

TWO  
displays table 2.

BOTH  
displays both table 1 and table 2.

Default  ONE
LOAD Statement

Loads a translation table into memory for editing.

**Tips:**
- Use the LOAD statement when you specify an incorrect table name in the PROC TRANTAB statement. You can specify the correct name without reinvoking the procedure.
- Use the LOAD statement to edit multiple translation tables in a single PROC TRANTAB step. (Be sure to save the first table before you load another one.)

**Syntax**

```
LOAD TABLE=table-name <NLS>;
```

**Required Argument**

```
TABLE=table-name
```

specifies the name of an existing translation table to edit. The specified table name must be a valid one-level SAS name.

**Optional Argument**

```
NLS
```

specifies that the table listed in the TABLE= argument is one of five special internal translation tables that are provided with SAS. You must use the NLS option when you specify one of the five special tables in the TABLE= argument:

- **SASXPT**
  - specifies the local-to-transport format translation table.
- **SASLCL**
  - specifies the transport-to-local format translation table.
- **SASUCS**
  - specifies the lowercase-to-uppercase translation table.
- **SASLCS**
  - specifies the uppercase-to-lowercase translation table.
- **SASCCL**
  - specifies the character classification table, which contains flag bytes that correspond to each character position. These positions indicate the class or classes to which each character belongs.

NLS stands for National Language Support. This option and the associated translation tables provide a method to map characters from languages other than English to programs, displays, and files. When you load one of these special translation tables, the SAS log displays a note that states that table 2 is uninitialized. That is, table 2 is an empty table that contains all 0s. PROC TRANTAB does not use table 2 for translation in these special cases.

REPLACE Statement

Replaces characters in a translation table with the specified values, starting at the specified position.
**Syntax**

REPLACE position value-1 < ...value-n>;

**Required Arguments**

*position*

specifies the position in a translation table where the replacement begins. The editable positions in a translation table begin at decimal 0 and end at decimal 255. To specify the position, you can do either of the following:

- Use a decimal or hexadecimal value to specify an actual location. If you specify a decimal value (for example, 20), PROC TRANTAB locates position 20 in the table, which is byte 21. If you specify a hexadecimal value (for example, '14'x), PROC TRANTAB locates the decimal position that is equivalent to the specified hexadecimal value, which in this case is position 20 (or byte 21) in the table.

- Use a quoted character. PROC TRANTAB locates the quoted character in the table (that is, the quoted character's hexadecimal value) and uses that character's position as the starting position. For example, specifying the following REPLACE statement replaces the first occurrence of the hexadecimal value for "a" and the next two hexadecimal values with the hexadecimal equivalent of "ABC": replace 'a' 'ABC';

This action is useful when you want to locate alphabetic and numeric characters, but you do not know their actual location. If the quoted character is not found, PROC TRANTAB displays an error message and ignores the statement.

To edit positions 256 through 511 (table 2), follow these steps:

1. Issue the SWAP statement.
2. Issue the appropriate REPLACE statement.
3. Issue the SWAP statement again to reposition the table.

*value-1 < ...value-n>*

is one or more decimal, hexadecimal, or character constants that specify the actual value to place in the table, starting at position. You can also use a mixture of these values. That is, you can specify a decimal, a hexadecimal, and a character value in one REPLACE statement. “Example 3: Editing By Specifying a Decimal Value for the Starting Position” on page 682 shows a mixture of all three types of values in the REPLACE statement.

---

**SAVE Statement**

Saves the translation table in your Sasuser.Profile catalog.

**Syntax**

SAVE <TABLE=table-name> <CAT=libname.catalog> <ONE | TWO | BOTH>;
**Optional Arguments**

**TABLE=table-name**  
specifies the name under which the current table is saved. The name must be a valid one-level SAS name.

**Default**  
If you omit the TABLE= option, the current table is saved under the name you specify in the PROC TRANTAB statement or in the LOAD statement.

**CAT=libname.catalog**  
specifies the name of the output catalog when you are saving a TRANTAB entry.

**ONE | TWO | BOTH**  

**ONE**  
saves table 1.

**TWO**  
saves table 2.

**BOTH**  
saves both table 1 and table 2.

**Default**  
BOTH

---

**SWAP Statement**

Exchanges table 1 with table 2 to enable you to edit positions 256 through 511.

**Tip:**  
After you edit the table, you must issue the SWAP statement again to reposition the table.

**Syntax**

SWAP;

---

**Examples: TRANTAB Procedure**

---

**Example 1: Viewing a Translation Table**

**Features:** LIST statement

This example uses PROC TRANTAB to display the ASCII translation table supplied by SAS. All examples were produced in the UNIX environment.

**Set the options and specify a translation table.**

```plaintext
options nodatepageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;
```
Example 2: Creating a Translation Table

Features:
- Procedures features:
  - LIST statement
  - REPLACE statement
  - SAVE statement

---

Display both halves of the translation table. The LIST BOTH statement displays both the table that provides the translation and the table that provides the inverse translation.

```
list both;
```

SAS Log

NOTE: Table specified is ASCII.
ASCII table 1:
```
0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '0010203040506070809A0B0C0D0E0F'x
10 '101112131415161718191A1B1C1D1E1F'x
20 '202122232425262728292A2B2C2D2E2F'x
30 '303132333435363738393A3B3C3D3E3F'x
40 '404142434445464748494A4B4C4D4E4F'x
50 '505152535455565758595A5B5C5D5E5F'x
60 '606162636465666768696A6B6C6D6E6F'x
70 '707172737475767778797A7B7C7D7E7F'x
80 '808182838485868788898A8B8C8D8E8F'x
90 '909192939495969798999A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACAABAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BBBCBCCDBEBBF'x
C0 'CC01CC03CC05CC07CC09CC0BCC0DCC0F'x
D0 'DD01DD03DD05DD07DD09DD0BDD0DDEEF'x
E0 'EE01EE03EE05EE07EE09EE0BEE0DEE0F'x
F0 'FF01FF03FF05FF07FF09FF0BFF0DFF0F'x
```

ASCII table 2:
```
0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '0010203040506070809A0B0C0D0E0F'x
10 '101112131415161718191A1B1C1D1E1F'x
20 '202122232425262728292A2B2C2D2E2F'x
30 '303132333435363738393A3B3C3D3E3F'x
40 '404142434445464748494A4B4C4D4E4F'x
50 '505152535455565758595A5B5C5D5E5F'x
60 '606162636465666768696A6B6C6D6E6F'x
70 '707172737475767778797A7B7C7D7E7F'x
80 '808182838485868788898A8B8C8D8E8F'x
90 '909192939495969798999A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACAABAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BBBCBCCDBEBBF'x
C0 'CC01CC03CC05CC07CC09CC0BCC0DCC0F'x
D0 'DD01DD03DD05DD07DD09DD0BDD0DDEEF'x
E0 'EE01EE03EE05EE07EE09EE0BEE0DEE0F'x
F0 'FF01FF03FF05FF07FF09FF0BFF0DFF0F'x
```

Example 2: Creating a Translation Table
This example uses PROC TRANTAB to create a customized translation table. All examples were produced in the UNIX environment.

**Set the system options and specify the translation table to edit.**

```plaintext
options nodate pageno=1 linesize=80
    pagesize=60;
proc trantab table=newtable;
```

**Replace characters in the translation table starting at a specified position.** The REPLACE statement places the values in the table starting at position 0. You can use hexadecimal values of any length in the REPLACE statement. This example uses strings of length 16 to match how translation tables appear in the SAS log.

```plaintext
replace 0
    '00010203a309e5ff9ec40b0c0d0e0f'x
    '10111213a5e08e71819c6c51c1d1e1f'x
    'c7fcee92e40a171beaeb8eefe050607'x
    'c9e616f4f6f2f04ff6d8dca2b6a7501a'x
    '20e1edf3faf1d1aababfa22e3c282b7c'x
    '265facbdca1abb5f5f21242a293bac'x
    '2d2f5fa6a6a62b2ba6a62c255f3e3f'x
    'a62b2b2b2b2b22d2d2603a2340273d22'x
    '2b6162636465666768692d2ba6a62b2b'x
    '2d6a6b6c6d6e6f7071722da62d22d2d'x
    '2d7e737457677787a2d2b2b2b2b2b'x
    '2b2b2b5ff6a65f5f5ff65f5ff5b5f'x
    '7b4a4b4c4d4e4f5051525f5f5f5f'x
    '7d4a4b4c4d4e4f5051525f5f5f5f5f'x
    '5c835355565758595a5ff75f5f5f'x
    '30313233343536373839b75f6eb25f5f'x
;
```

**Save the table.** The SAVE statement saves the table under the name that is specified in the PROC TRANTAB statement. By default, the table is saved in your Sasuser.Profile catalog.

```plaintext
save;
```

**Display both halves of the translation table in the SAS log.** The LIST BOTH statement displays both the table that provides the translation and the table that provides the inverse translation.

```plaintext
list both;
```
Example 2: Creating a Translation Table

--- Create and edit table 2. Table 2 is empty; that is, it consists entirely of 0s. To create table 2, you can use the INVERSE statement. (See: SAS(R) Statements: Reference)

To edit table 2, you can use the SWAP statement with the REPLACE statement. (See: SAS(R) Statements: Reference)

NOTE: Table specified is NEWTABLE.
WARNING: Table NEWTABLE not found! New table is assumed.
NOTE: NEWTABLE table 1 is uninitialized.
NOTE: NEWTABLE table 2 is uninitialized.

NOTE: Saving table NEWTABLE.
NOTE: NEWTABLE table 2 will not be saved because it is uninitialized.

NEWTABLE table 1:

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'00010203A309E57FF9ECC40B0C0D0E0F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'10111213A5E008718199C6C51C1D1E1F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'C7FCE92E840A1718BE88BFE050607'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'C9B61E74F6F2FB04F8ED0CA2B6A7501A'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'20E1EDF3F6AF1DA2E1C2E2B7C'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'265FACBDEA1ABB5F5F21242A931BAC'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>'2D2F5A6A66A62B2A6A62C525E3F3F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'2861235645565667666592D2A6A62E2B'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'2D6A6BCE6D6F7071722D62D22D2D2D'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'2B2B2B5F6A5F5F5F5F65F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'784124344445464748495F5F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'7D4AA4B4C4D4F4F5051525F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'5C835354555675859A5F5F5F5F5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'30311333343536373839B56EB2B5F5F'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NEWTABLE table 2:

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>'00000000000000000000000000000000'x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 3: Editing By Specifying a Decimal Value for the Starting Position

Features:
- LIST statement
- REPLACE statement
- SAVE statement

This example edits the translation table that was created in “Example 2: Creating a Translation Table” on page 679. The decimal value specified in the REPLACE statement marks the starting position for the table changes.

The vertical arrow in both SAS logs marks the point at which the changes begin.

All examples were produced in the UNIX environment.

Program 1: Display the Original Table

```sas
options nodate pageno=1 linesize=80 pagesize=60; proc trantab table=newtable;
list one;
```

Program Description

Set the system options and specify the translation table to edit.

```sas
options nodate pageno=1 linesize=80 pagesize=60; proc trantab table=newtable;
```

Display the original table. This LIST statement displays the original NEWTABLE translation table.

```sas
list one;
```

SAS Log

Original NEWTABLE Translation Table
Table specified is NEWTABLE.
NOTE: NEWTABLE table 2 is uninitialized.

NEWTABLE table 1:

```
  0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '00010203A309E57FF9ECC40B0C0D0E0F'x
10 '1011213A58B08E71819C6C51C1D1E1F'x
20 'C7FCE9E240A171BEE88B8EFE05607'x
30 'C9E616F4F3F2FB04FDF6DCA2B6A7501A'x
40 '20E1EDF3FAF1D1AABB8FA222E3C282B7C'x
50 '265FA6CD8C113BB85F5F2124A293BAC'x
60 '202F566A6A622BBA62C255F3E3F'x
80 '2B6162636465666768692D2B2A6A62B2B'x
90 '2D6A6B6C6D6E6F7071722DA62D2B2D2D'x
B0 '2B22B5F5FA65F5F5FDF5F6F5F5F5F5F'x
C0 '7B4142434445464748495F5F5F5F5F5F'x
D0 '7D4A4B4C4D4E4F5051525F5F5F5F5F5F'x
E0 '5C83535455565758595A5F5F5F75F5F5B0'x
F0 '303132333435363738393B75F6EB255F5F'x
```

Program 2: Edit the Table

```
replace 10
  20 10 200 'x' 'ux' '092040'x;

save;
list one;
```

Program Description

Replace characters in the translation table, starting at a specified position. The REPLACE statement starts at position decimal 10, which is byte 11 in the original table, and performs a byte-to-byte replacement with the given values.

```
replace 10
  20 10 200 'x' 'ux' '092040'x;
```

Save the changes. The SAVE statement saves the changes that you made to the NEWTABLE translation table.

```
save;
```

Display the new table. The second LIST statement displays the edited NEWTABLE translation table.

```
list one;
```
SAS Log

```
Saving table NEWTABLE.
NOTE: NEWTABLE table 2 will not be saved because it is uninitialized.
NEWTABLE table 1:
↓
  0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '00010203A309E57FF9EC140AC8787578'x
10 '09204013A5E008E71819C6C51C1D1E1F'x
20 'C7FCE9E2E40A171BE1E8EFE050607'x
30 'C9E616F4F6F204FDF6DCA2B6A7501A'x
40 '20E1EDF3FAF1D1AABA8FA22B3C282B7C'x
50 '265FACE9BCA1ABB5F5F21242A293BAC'x
60 '2D2F5F666A62B2B6A622C255F3E3F'x
80 '2B612636E666666666662D23A6A62B2'x
90 '2D646666666666666666666662D2B2D2D'x
A0 '2D7E73475767778787A2D2B2B2B2B2'x
B0 '2B2B2B5F5F5F5F5F5F5F5F5F5F5F5F'x
C0 '7B41424344454647484955F5F5F5F5F'x
D0 '7D4A4B4C4D4E4F5555555555555555'x
E0 '5C8353545555555555555555555555'x
F0 '303132333435363738393B75F6EB25F5F'x
```

Output Details

At position 10 (which is byte 11), a vertical arrow denotes the starting point for the changes to the translation table.
At byte 11, decimal 20 (which is hexadecimal 14) replaces hexadecimal C4.
At byte 12, decimal 10 (which is hexadecimal 0A) replaces hexadecimal 0B.
At byte 13, decimal 200 (which is hexadecimal C8) replaces hexadecimal 0C.
At byte 14, character 'x' (which is hexadecimal 78) replaces hexadecimal 0D.
At bytes 15 and 16, characters 'ux' (which are hexadecimal 75 and 78, respectively) replace hexadecimal 0E and 0F.
At bytes 17, 18, and 19, hexadecimal 092040 replaces hexadecimal 101112.

Example 4: Editing By Using a Quoted Character for the Starting Position

```
Features: LIST statement
          LOAD statement
          REPLACE statement
          SAVE statement

This example creates a new translation table by editing the ASCII translation table. The first occurrence of the hexadecimal equivalent of the quoted character that was specified in the REPLACE statement is the starting position for the changes to the table. This method differs from “Example 3: Editing By Specifying a Decimal Value for the Starting Position” on page 682 because you do not need to know the exact position at which to start the changes to the table. PROC TRANTAB finds the correct position for you.

The edited table is saved under a new name. Horizontal arrows in both SAS logs denote the edited rows in the translation table.
```
All examples were produced in the UNIX environment.

**Program 1: Display the Original Table**

```sas
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;
list one;
```

**Program Description**

Set the system options and specify which translation table to edit.

```sas
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;
```

Display the translation table. The LIST statement displays the original translation table in the SAS log.

```sas
list one;
```

**SAS Log**

```
NOTE: Table specified is ASCII.
ASCII table 1:
  0 1 2 3 4 5 6 7 8 9 A B C D E F
  00 '000102030405060708090A0B0C0D0E0F'x
  10 '101112131415161718191A1B1C1D1E1F'x
  20 '202122232425262728292A2B2C2D2E2F'x
  30 '303132333435363738393A3B3C3D3E3F'x
  40 '404142434445464748494A4B4C4D4E4F'x
  50 '505152535455565758595A5B5C5D5E5F'x
  60 '606162636465666768696A6B6C6D6E6F'x
  70 '707172737475767778797A7B7C7D7E7F'x
  80 '808182838485868788898A8B8C8D8E8F'x
  90 '90919293949596979899A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
  A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
  B0 'B0B1B2B3B4B5B6B7B8B9BBBCCBDBEBEF'x
  C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
  D0 'D0D1D2D3D4D5D6D7D8D9DADBDCCDDEDF'x
  E0 'E0E1E2E3E4E5E6E7E8E9EAEFEDFEEF'x
  F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFEFF'x
```

**Program 2: Edit the Table**

```sas
replace 'a' 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
save table=upper;
load table=upper;
list one;
```

**Program Description**

Replace characters in the translation table, starting at a specified position. The REPLACE statement finds the first occurrence of the hexadecimal "a" (which is 61) and replaces it and the next 25 hexadecimal values with the hexadecimal values for uppercase "A" through "Z."
Save your changes. The SAVE statement saves the changes made to the ASCII translation table under the new table name UPPER. The stored contents of the ASCII translation table remain unchanged.

```
save table=upper;
```

Load and display the translation table. The LOAD statement loads the edited translation table UPPER. The LIST statement displays the translation table UPPER in the SAS log.

```
load table=upper;
list one;
```

### Example 5: Creating the Inverse of a Table

**Features:**
- INVERSE statement
- LIST statement
- SAVE statement

This example creates the inverse of the translation table that was created in “Example 4: Editing By Using a Quoted Character for the Starting Position” on page 684. The new translation table that is created in this example is the operating environment-to-device translation for use in data communications.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=upper;
```

Create the inverse translation table, save the tables, and then display them. The INVERSE statement creates table 2 by inverting the original table 1 (called UPPER). The SAVE statement saves the translation tables. The LIST BOTH statement displays both the original translation table and its inverse.
SAS Log

The INVERSE statement lists in the SAS log all of the multiple translations that it encounters as it inverts the translation table. In “Example 4: Editing By Using a Quoted Character for the Starting Position” on page 684, all the lowercase letters are converted to uppercase in the translation table UPPER, which means that there are two sets of uppercase letters in UPPER. When INVERSE cannot make a translation, PROC TRANTAB fills the value with 00. Note that the inverse of the translation table UPPER has numerous 00 values.
The SAS log lists all the duplicate values that it encounters as it creates the inverse of table 1. To conserve space, most of these messages are deleted in this example.

```
NOTE: This table cannot be mapped one to one.
duplicate of '41'x found at '61'x in table one.
duplicate of '42'x found at '62'x in table one.
duplicate of '43'x found at '63'x in table one.
.
.
duplicate of '58'x found at '78'x in table one.
duplicate of '59'x found at '79'x in table one.
duplicate of '5A'x found at '7A'x in table one.
NOTE: Saving table UPPER.
UPPER table 1:
0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '000102030405060708090A0B0C0D0E0F'x
10 '101112131415161718191A1B1C1D1E1F'x
20 '202122232425262728292A2B2C2D2E2F'x
30 '303132333435363738393A3B3C3D3E3F'x
40 '404142434445464748494A4B4C4D4E4F'x
50 '505152535455565758595A5B5C5D5E5F'x
60 '606162636465666768696A6B6C6D6E6F'x
70 '707172737475767778797A7B7C7D7E7F'x
80 '808182838485868788898A8B8C8D8E8F'x
90 '90919293949596979899A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDEBF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADBCDDDEDF'x
E0 'E0E1E2E3E4E5E6E7E8E9E8ECEDEEEEF'x
F0 'F0F1F2F3F4F5F6F7F8F9F8F7F6F5F4F3F2F1F0'x
```

```
UPPER table 2:
0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '000102030405060708090A0B0C0D0E0F'x
10 '101112131415161718191A1B1C1D1E1F'x
20 '202122232425262728292A2B2C2D2E2F'x
30 '303132333435363738393A3B3C3D3E3F'x
40 '404142434445464748494A4B4C4D4E4F'x
50 '505152535455565758595A5B5C5D5E5F'x
60 '60000000000000000000000000000000'x
70 '00000000000000000000007B7C7D7E7F'x
80 '808182838485868788898A8B8C8D8E8F'x
90 '90919293949596979899A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDEBF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADBCDDDEDF'x
E0 'E0E1E2E3E4E5E6E7E8E9E8ECEDEEEEF'x
F0 'F0F1F2F3F4F5F6F7F8F9F8F7F6F5F4F3F2F1F0'x
```

Example 6: Using Different Translation Tables for Sorting

**Features:**

- PROC SORT statement option: SORTSEQ=
- PRINT procedure
This example shows how to specify a different translation table to sort data in an order that is different from the default sort order. Characters that are written in a language other than U.S. English might require a sort order that is different from the default order.

You can use the TRABASE program in the SAS Sample Library to create translation tables for several languages. All examples were produced in the UNIX environment.

---

**Set the SAS system options.**

```
options nodate pageno=1 linesize=80 pagesize=60;
```

---

**Create the TESTSORT data set.** The DATA step creates a SAS data set with four pairs of words, each pair different only in the case of the first letter.

```
data testsort;
  input Values $10.;
datalines;
Always
always
Forever
forever
Later
later
Yesterday
yesterday
;
```

---

**Sort the data in an order that is different from the default sort order.** PROC SORT sorts the data by using the default translation table, which sorts all lowercase words first, then all uppercase words.

```
proc sort;
  by values;
run;
```

---

**Print the data set.** PROC PRINT prints the sorted data set.

```
proc print noobs;
  title 'Default Sort Sequence';
run;
```
SAS Output

The following output is the output from sorting values with the default translation table. The default sort sequence sorts all the capitalized words in alphabetical order before it sorts any lowercase words.

<table>
<thead>
<tr>
<th>Default Sort Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>Values</td>
</tr>
<tr>
<td>Always</td>
</tr>
<tr>
<td>Forever</td>
</tr>
<tr>
<td>Later</td>
</tr>
<tr>
<td>Yesterday</td>
</tr>
<tr>
<td>always</td>
</tr>
<tr>
<td>forever</td>
</tr>
<tr>
<td>later</td>
</tr>
<tr>
<td>yesterday</td>
</tr>
</tbody>
</table>

Sort the data according to the translation table UPPER and print the new data set. The SORTSEQ= system option specifies that PROC SORT sort the data according to the customized translation table UPPER, which treats lowercase and uppercase letters alike. This method is useful for sorting without regard for case. PROC PRINT prints the sorted data set.

```sas
proc sort sortseq=upper;
  by values;
run;
proc print noobs;
  title 'Customized Sort Sequence';
run;
```

SAS Output

The following output is the result from sorting values with a customized translation table. The customized sort sequence sorts all the words in alphabetical order, without regard for the case of the first letters.

<table>
<thead>
<tr>
<th>Customized Sort Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>Values</td>
</tr>
<tr>
<td>Always</td>
</tr>
<tr>
<td>always</td>
</tr>
<tr>
<td>Forever</td>
</tr>
<tr>
<td>forever</td>
</tr>
<tr>
<td>Later</td>
</tr>
<tr>
<td>later</td>
</tr>
<tr>
<td>Yesterday</td>
</tr>
<tr>
<td>yesterday</td>
</tr>
</tbody>
</table>

Example 7: Editing Table 1 and Table 2

Features: List statement
This example shows how to edit both areas of a translation table. To edit positions 256 through 511 (table 2), you must

- Issue the SWAP statement to have table 2 change places with table 1.
- Issue an appropriate REPLACE statement to make changes to table 2.
- Issue the SWAP statement again to reposition the table.

Arrows in the SAS logs mark the rows and columns that are changed.

**Set the SAS system options and specify the translation table.**

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=upper;
```

**Display the original translation table. The LIST statement displays the original UPPER translation table.**

```
list both;
```
SAS Log

The following output is the original UPPER translation table.

```
NOTE: Table specified is UPPER.
UPPER table 1:
  ↓
  0 1 2 3 4 5 6 7 8 9 A B C D E F
 0 '000102030405060708090A0B0C0D0E0F'x ←
 10 '10112131415161718191A1B1C1D1E1F'x
 20 '202122232425262728292A2B2C2D2E2F'x
 30 '303132333435363738393A3B3C3D3E3F'x
 40 '404142434445464748494A4B4C4D4E4F'x
 50 '505152535455565758595A5B5C5D5E5F'x
 60 '606162636465666768696A6B6C6D6E6F'x
 70 '707172737475767778797A7B7C7D7E7F'x
 80 '808182838485868788898A8B8C8D8E8F'x
 90 '909192939495969798999A9B9C9D9E9F'x
AO 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBBEBF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACBCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADDDEDFE'x
E0 'E0E1E2E3E4E5E6E7E8E9EAEFEDFEDF'x
F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

Replace characters in the translation table starting at a specified position. The REPLACE statement starts at position 1 and replaces the current value of 01 with '0A'.

```
replace 1 '0A'x;
```

Prepare table 2 for editing. The first SWAP statement positions table 2 so that it can be edited. The second REPLACE statement makes the same change in table 2 that was made in table 1.

```
swap;
replace 1 '0A'x;
```

Save and display the tables in their original positions. The second SWAP statement restores table 1 and table 2 to their original positions. The SAVE statement saves
both areas of the translation table by default. The LIST statement displays both areas of the table.

```
swap;
save;
list both;
```

**SAS Log**

The upper table 1 is modified. The hexadecimal value '0A' replaces hexadecimal value 01 in byte 2 for both areas of the translation table. Arrows mark the rows and columns of the table in which this change is made.

```
NOTE: Table specified is UPPER.
UPPER table 1:
[darr]
  0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '000A02030405060708090A0B0C0D0E0F'x <--
 10 '101112131415161718191A1B1C1D1E1F'x
 20 '202122232425262728292A2B2C2D2E2F'x
 30 '303132333435363738393A3B3C3D3E3F'x
 40 '404142434445464748494A4B4C4D4E4F'x
 50 '505152535455565758595A5B5C5D5E5F'x
 60 '606162636465666768696A6B6C6D6E6F'x
 70 '707172737475767778797A7B7C7D7E7F'x
 80 '808182838485868788898A8B8C8D8E8F'x
 90 '909192939495969798999A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBBEBEF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADBDDEDDDF'x
E0 'E0E1E2E3E4E5E6E7E8E9EAECECEDEEEF'x
F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFFFF'x

UPPER table 2:
[darr]
  0 1 2 3 4 5 6 7 8 9 A B C D E F
00 '000A02030405060708090A0B0C0D0E0F'x <--
 10 '101112131415161718191A1B1C1D1E1F'x
 20 '202122232425262728292A2B2C2D2E2F'x
 30 '303132333435363738393A3B3C3D3E3F'x
 40 '404142434445464748494A4B4C4D4E4F'x
 50 '505152535455565758595A5B5C5D5E5F'x
 60 '60000000000000000000000000000000'x
 70 '00000000000000000000000000000000'x
 80 '808182838485868788898A8B8C8D8E8F'x
 90 '909192939495969798999A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBBEBEF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADBDDEDDDF'x
E0 'E0E1E2E3E4E5E6E7E8E9EAECECEDEEEF'x
F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFFFF'x
```
Part 11

Values for Locale, Encoding, and Transcoding

Chapter 20
Values for the LOCALE= System Option .................................. 697

Chapter 21
SAS System Options for Processing DBCS Data ......................... 711

Chapter 22
Encoding Values in SAS Language Elements ............................. 713

Chapter 23
Encoding Values for a SAS Session ....................................... 723
Chapter 20
Values for the LOCALE= System Option

LOCALE= Values and Default Settings for ENCODING, PAPERSIZE, DFLANG, and DATESTYLE Options

The following table lists the valid LOCALE= values, specified by using the SAS name or the POSIX name. The alias name is also listed. Some locales do not have an alias.

Table 20.1  Values for the LOCALE= System Option

<table>
<thead>
<tr>
<th>SAS Name</th>
<th>POSIX Locale</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans_SouthAfrica</td>
<td>af_ZA</td>
<td>Afrikaans</td>
</tr>
<tr>
<td></td>
<td></td>
<td>af</td>
</tr>
<tr>
<td>Albanian_Albania</td>
<td>sq_AL</td>
<td>Albanian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sq</td>
</tr>
<tr>
<td>Arabic_Algeria</td>
<td>ar_DZ</td>
<td></td>
</tr>
<tr>
<td>Arabic_Bahrain</td>
<td>ar_BH</td>
<td></td>
</tr>
<tr>
<td>Arabic_Egypt</td>
<td>ar_EG</td>
<td></td>
</tr>
<tr>
<td>Arabic_India</td>
<td>ar_IN</td>
<td></td>
</tr>
<tr>
<td>Arabic_Iraq</td>
<td>ar_IQ</td>
<td></td>
</tr>
<tr>
<td>Arabic_Jordan</td>
<td>ar_JO</td>
<td></td>
</tr>
<tr>
<td>Arabic_Kuwait</td>
<td>ar_KW</td>
<td></td>
</tr>
<tr>
<td>Arabic_Lebanon</td>
<td>ar_LB</td>
<td></td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Arabic_Libya</td>
<td>ar_LY</td>
<td></td>
</tr>
<tr>
<td>Arabic_Morocco</td>
<td>ar_MA</td>
<td></td>
</tr>
<tr>
<td>Arabic_Oman</td>
<td>ar_OM</td>
<td></td>
</tr>
<tr>
<td>Arabic_Qatar</td>
<td>ar_QA</td>
<td></td>
</tr>
<tr>
<td>Arabic_SaudiArabia</td>
<td>ar_SA</td>
<td></td>
</tr>
<tr>
<td>Arabic_Sudan</td>
<td>ar_SD</td>
<td></td>
</tr>
<tr>
<td>Arabic_Syria</td>
<td>ar_SY</td>
<td></td>
</tr>
<tr>
<td>Arabic_Tunisia</td>
<td>ar_TN</td>
<td></td>
</tr>
<tr>
<td>Arabic_UnitedArabEmirates</td>
<td>ar_AE</td>
<td>Arabic ar</td>
</tr>
<tr>
<td>Arabic_Yemen</td>
<td>ar_YE</td>
<td></td>
</tr>
<tr>
<td>Basque_Spain</td>
<td>eu_ES</td>
<td>Basque Euskara eu</td>
</tr>
<tr>
<td>Bengali_India</td>
<td>bn_IN</td>
<td>Bengali bn</td>
</tr>
<tr>
<td>Bosnian_BosniaHerzegovina</td>
<td>bs_BA</td>
<td>Bosnian bs</td>
</tr>
<tr>
<td>Bulgarian_Bulgaria</td>
<td>bg_BG</td>
<td>Bulgarian bg</td>
</tr>
<tr>
<td>Byelorussian_Belarus</td>
<td>be_BY</td>
<td>Byelorussian Belarus Byelorussian_Belarus be</td>
</tr>
<tr>
<td>Catalan_Spain</td>
<td>ca_ES</td>
<td>Catalan ca</td>
</tr>
<tr>
<td>Chinese_China</td>
<td>zh_CN</td>
<td>Chinese zh</td>
</tr>
<tr>
<td>Chinese_HongKong</td>
<td>zh_HK</td>
<td></td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>Chinese_Macau</td>
<td>zh_MO</td>
<td></td>
</tr>
<tr>
<td>Chinese_Singapore</td>
<td>zh_SG</td>
<td></td>
</tr>
<tr>
<td>Chinese_Taiwan</td>
<td>zh_TW</td>
<td></td>
</tr>
<tr>
<td>Cornish_UnitedKingdom</td>
<td>kw_GB</td>
<td>Cornish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kw</td>
</tr>
<tr>
<td>Croatian_BosniaHerzegovina</td>
<td>hr_BA</td>
<td></td>
</tr>
<tr>
<td>Croatian_Croatia</td>
<td>hr_HR</td>
<td>Croatian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hr</td>
</tr>
<tr>
<td>Czech_CzechRepublic</td>
<td>cs_CZ</td>
<td>Czech</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cs</td>
</tr>
<tr>
<td>Danish_Denmark</td>
<td>da_DK</td>
<td>Danish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>da</td>
</tr>
<tr>
<td>Dutch_Belgium</td>
<td>nl_BE</td>
<td></td>
</tr>
<tr>
<td>Dutch_Netherlands</td>
<td>nl_NL</td>
<td>Dutch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nl</td>
</tr>
<tr>
<td>English_Australia</td>
<td>en_AU</td>
<td></td>
</tr>
<tr>
<td>English_Belgium</td>
<td>en_BE</td>
<td></td>
</tr>
<tr>
<td>English_Botswana</td>
<td>en_BW</td>
<td></td>
</tr>
<tr>
<td>English_Canada</td>
<td>en_CA</td>
<td></td>
</tr>
<tr>
<td>English_Caribbean</td>
<td>en_CB</td>
<td></td>
</tr>
<tr>
<td>English_HongKong</td>
<td>en_HK</td>
<td></td>
</tr>
<tr>
<td>English_India</td>
<td>en_IN</td>
<td></td>
</tr>
<tr>
<td>English_Ireland</td>
<td>en_IE</td>
<td></td>
</tr>
<tr>
<td>English_Jamaica</td>
<td>en_JM</td>
<td></td>
</tr>
<tr>
<td>English_Malta</td>
<td>en_MT</td>
<td></td>
</tr>
<tr>
<td>English_NewZealand</td>
<td>en_NZ</td>
<td></td>
</tr>
<tr>
<td>English_Philippines</td>
<td>en_PH</td>
<td></td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>English_Singapore</td>
<td>en_SG</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>en</td>
</tr>
<tr>
<td>English_SouthAfrica</td>
<td>en_ZA</td>
<td></td>
</tr>
<tr>
<td>English_UnitedKingdom</td>
<td>en_GB</td>
<td></td>
</tr>
<tr>
<td>English_UnitedStates</td>
<td>en_US</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td></td>
<td>en</td>
</tr>
<tr>
<td>English_Zimbabwe</td>
<td>en_ZW</td>
<td></td>
</tr>
<tr>
<td>Estonian_Estonia</td>
<td>et_EE</td>
<td>Estonian</td>
</tr>
<tr>
<td></td>
<td></td>
<td>et</td>
</tr>
<tr>
<td>Faroese_FaroeIslands</td>
<td>fo_FO</td>
<td>Faroese</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fo</td>
</tr>
<tr>
<td>Finnish_Finland</td>
<td>fi_FI</td>
<td>Finnish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fi</td>
</tr>
<tr>
<td>French_Belgium</td>
<td>fr_BE</td>
<td></td>
</tr>
<tr>
<td>French_Canada</td>
<td>fr_CA</td>
<td></td>
</tr>
<tr>
<td>French_France</td>
<td>fr_FR</td>
<td>French</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fr</td>
</tr>
<tr>
<td>French_Luxembourg</td>
<td>fr_LU</td>
<td></td>
</tr>
<tr>
<td>French_Switzerland</td>
<td>fr_CH</td>
<td></td>
</tr>
<tr>
<td>German_Austria</td>
<td>de_AT</td>
<td></td>
</tr>
<tr>
<td>German_Germany</td>
<td>de_DE</td>
<td>German</td>
</tr>
<tr>
<td></td>
<td></td>
<td>de</td>
</tr>
<tr>
<td>German_Liechtenstein</td>
<td>de_LI</td>
<td></td>
</tr>
<tr>
<td>German_Luxembourg</td>
<td>de_LU</td>
<td></td>
</tr>
<tr>
<td>German_Switzerland</td>
<td>de_CH</td>
<td></td>
</tr>
<tr>
<td>Greek_Cyprus</td>
<td>el_CY</td>
<td></td>
</tr>
<tr>
<td>Greek_Greece</td>
<td>el_GR</td>
<td>Greek</td>
</tr>
<tr>
<td></td>
<td></td>
<td>el</td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Greenlandic_Greenland</td>
<td>kl_GL</td>
<td>Greenlandic kl</td>
</tr>
<tr>
<td>Hebrew_Israel</td>
<td>he_IL</td>
<td>Hebrew he</td>
</tr>
<tr>
<td>Hindi_India</td>
<td>hi_IN</td>
<td>Hindi hi</td>
</tr>
<tr>
<td>Hungarian_Hungary</td>
<td>hu_HU</td>
<td>Hungarian hu</td>
</tr>
<tr>
<td>Icelandic_Iceland</td>
<td>is_IS</td>
<td>Icelandic is</td>
</tr>
<tr>
<td>Indonesian_Indonesia</td>
<td>id_ID</td>
<td>Indonesian id</td>
</tr>
<tr>
<td>Irish_Ireland</td>
<td>ga_IE</td>
<td>Irish ga</td>
</tr>
<tr>
<td>Italian_Italy</td>
<td>it_IT</td>
<td>Italian it</td>
</tr>
<tr>
<td>Italian_Switzerland</td>
<td>it_CH</td>
<td></td>
</tr>
<tr>
<td>Japanese_Japan</td>
<td>ja_JP</td>
<td>Japanese ja</td>
</tr>
<tr>
<td>Kazakh_Kazakhstan</td>
<td>kk_KZ</td>
<td>Kazakh kk</td>
</tr>
<tr>
<td>Korean_Korea</td>
<td>ko_KR</td>
<td>Korean ko</td>
</tr>
<tr>
<td>Latvian_Latvia</td>
<td>lv_LV</td>
<td>Latvian lv</td>
</tr>
<tr>
<td>Lithuanian_Lithuania</td>
<td>lt_LT</td>
<td>Lithuanian lt</td>
</tr>
<tr>
<td>Macedonian_Macedonia</td>
<td>mk_MK</td>
<td>Macedonian mk</td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Malay_Malaysia</td>
<td>ms_MY</td>
<td>Malay, ms</td>
</tr>
<tr>
<td>Maltese_Malta</td>
<td>mt_MT</td>
<td>Maltese, mt</td>
</tr>
<tr>
<td>Marathi_India</td>
<td>mr_IN</td>
<td>Marathi, mr</td>
</tr>
<tr>
<td>NorwegianBokmal_Norway</td>
<td>nb_NO</td>
<td>NorwegianBokmal, nb</td>
</tr>
<tr>
<td>NorwegianNynorsk_Norway</td>
<td>nn_NO</td>
<td>NorwegianNynorsk, nn</td>
</tr>
<tr>
<td>Norwegian_Norway</td>
<td>no_NO</td>
<td>Norwegian, no</td>
</tr>
<tr>
<td>Persian_India</td>
<td>fa_IN</td>
<td>Persian, fa</td>
</tr>
<tr>
<td>Persian_Iran</td>
<td>fa_IR</td>
<td>Persian, fa</td>
</tr>
<tr>
<td>Polish_Poland</td>
<td>pl_PL</td>
<td>Polish, pl</td>
</tr>
<tr>
<td>Portuguese_Brazil</td>
<td>pt_BR</td>
<td>Portuguese, pt</td>
</tr>
<tr>
<td>Portuguese_Portugal</td>
<td>pt_PT</td>
<td>Portuguese, pt</td>
</tr>
<tr>
<td>Romanian_Romania</td>
<td>ro_RO</td>
<td>Romanian, ro</td>
</tr>
<tr>
<td>Russian_Russia</td>
<td>ru RU</td>
<td>Russian, ru</td>
</tr>
<tr>
<td>Russian_Ukraine</td>
<td>ru_UA</td>
<td>Russian, ru</td>
</tr>
<tr>
<td>Serbian_BosniaHerzegovina</td>
<td>sr_BA</td>
<td>Serbian, sr</td>
</tr>
<tr>
<td>Serbian_Montenegro</td>
<td>sr_ME</td>
<td>Serbian, sr</td>
</tr>
<tr>
<td>Serbian_Serbia</td>
<td>sr_RS</td>
<td>Serbian, sr</td>
</tr>
<tr>
<td>SerbianLatin_BosniaHerzegovina</td>
<td>sh_BA</td>
<td></td>
</tr>
<tr>
<td>SAS Name</td>
<td>POSIX Locale</td>
<td>Aliases</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>SerbianLatin_Montenegro</td>
<td>sh_ME</td>
<td></td>
</tr>
<tr>
<td>SerbianLatin_Serbia</td>
<td>sh_RS</td>
<td>SerbianLatin</td>
</tr>
<tr>
<td>Slovak_Slovakia</td>
<td>sk_SK</td>
<td>Slovak</td>
</tr>
<tr>
<td>Slovakian_Slovakia</td>
<td>sk</td>
<td>Slovakian_Slovakia</td>
</tr>
<tr>
<td>Slovenian_Slovenia</td>
<td>sl_SI</td>
<td>Slovenian</td>
</tr>
<tr>
<td>Spanish_Argentina</td>
<td>es_AR</td>
<td></td>
</tr>
<tr>
<td>Spanish_Bolivia</td>
<td>es_BO</td>
<td></td>
</tr>
<tr>
<td>Spanish_Chile</td>
<td>es_CL</td>
<td></td>
</tr>
<tr>
<td>Spanish_Colombia</td>
<td>es_CO</td>
<td></td>
</tr>
<tr>
<td>Spanish_CostaRica</td>
<td>es_CR</td>
<td></td>
</tr>
<tr>
<td>Spanish_DominicanRepublic</td>
<td>es_DO</td>
<td></td>
</tr>
<tr>
<td>Spanish_Ecuador</td>
<td>es_EC</td>
<td></td>
</tr>
<tr>
<td>Spanish_ElSalvador</td>
<td>es_SV</td>
<td></td>
</tr>
<tr>
<td>Spanish_Guatemala</td>
<td>es_GT</td>
<td></td>
</tr>
<tr>
<td>Spanish_Honduras</td>
<td>es_HN</td>
<td></td>
</tr>
<tr>
<td>Spanish_Mexico</td>
<td>es_MX</td>
<td></td>
</tr>
<tr>
<td>Spanish_Nicaragua</td>
<td>es_NI</td>
<td></td>
</tr>
<tr>
<td>Spanish_Panama</td>
<td>es_PA</td>
<td></td>
</tr>
<tr>
<td>Spanish_Paraguay</td>
<td>es_PY</td>
<td></td>
</tr>
<tr>
<td>Spanish_Peru</td>
<td>es_PE</td>
<td></td>
</tr>
<tr>
<td>Spanish_PuertoRico</td>
<td>es_PR</td>
<td></td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>es_ES</td>
<td>Spanish</td>
</tr>
</tbody>
</table>
The following table lists the valid POSIX values and the default settings for the ENCODING= option, by operating environment. The settings for DFLANG, DATESTYLE, and PAPERSIZE system options are set automatically.

Here is an example:

```bash
sas9 -locale arabic_algeria
```

When the Arabic_Algeria LOCALE= value is specified, corresponding default settings for the system options are as follows:

- DFLANG=English
- DATESTYLE=DMY
- PAPERSIZE=A4

### Table 20.2 Default Values for the ENCODING, DFLANG, DATESTYLE, and PAPERSIZE System Options Based on the LOCALE= System Option

<table>
<thead>
<tr>
<th>POSIX Locale</th>
<th>Windows Encoding</th>
<th>UNIX Encoding</th>
<th>z/OS Encoding</th>
<th>DFLANG=</th>
<th>DATESTYLE=</th>
<th>PAPERSIZE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>af_ZA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ar_AE</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>---------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ar_BH</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_DZ</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_EG</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IQ</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_JO</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_KW</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_LB</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_LY</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_MA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_OM</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_QA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SD</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SY</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_TN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_YE</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>be_BY</td>
<td>wecyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>bg_BG</td>
<td>wecyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>bn_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>bs_BA</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ca_ES</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>cs_CZ</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Czech</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>da_DK</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Danish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_AT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>Swiss_German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_DE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPER SIZE=</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>de_LI</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_LU</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>el_CY</td>
<td>Wgreek</td>
<td>greek</td>
<td>Open_ed-875</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>el_GR</td>
<td>wgreek</td>
<td>greek</td>
<td>open_ed-875</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_AU</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BW</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_CA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>en_CB</td>
<td>wlatin</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>en_GB</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_HK</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IN</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_JM</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_MT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>Open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_NZ</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_PH</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>en_SG</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>en_US</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>en_ZA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>en_ZW</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>es_AR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_BO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_CL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_DO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_EC</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>POSIX Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>----------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>es_ES</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1145</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_GT</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_HN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>es_MX</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_NI</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_PR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_SV</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_US</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_UY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>es_VE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>et_EE</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-1122</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>eu_ES</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed_1145</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-1097</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IR</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-1097</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fi_FI</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1143</td>
<td>Finnish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fo_FO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_BE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_CA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>French</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>fr_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>Swiss_French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_FR</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1147</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr-LU</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1147</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ga_IE</td>
<td>Wlatin1</td>
<td>Latin8</td>
<td>Open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>gv_GB</td>
<td>wlatin1</td>
<td>latin8</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>he_IL</td>
<td>whebrew</td>
<td>hebrew</td>
<td>open_ed-424</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>hi_IN</td>
<td>pciscii806</td>
<td>latin1</td>
<td>open_ed-1137</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
</tbody>
</table>
| POSIX Locale | Windows Encoding | UNIX Encoding | z/OS Encoding | DFLANG= | DATESTYLE= | PAGE SIZE=
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hr_BA</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Croatian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>hr_HR</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Croatian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>hu_HU</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Hungarian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>id_ID</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>is_IS</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_IT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1144</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ja_JP</td>
<td>shift-jis</td>
<td>pciscii806</td>
<td>ibm-939</td>
<td>Japanese</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kk_KZ</td>
<td>kz1048</td>
<td>Cyrillic</td>
<td>open_ed-1166</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>kl_GL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ko_KR</td>
<td>euc-kr</td>
<td>euc-kr</td>
<td>ibm-933</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kw_GB</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>lt_LT</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-1112</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>lv_LV</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-1112</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>mk_MK</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1154</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>mr_IN</td>
<td>pciscii806</td>
<td>latin1</td>
<td>open_ed-1137</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ms_MY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>mt_MT</td>
<td>wlatin1</td>
<td>latin3</td>
<td>open_ed-905</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nb_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_BE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1148</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_NL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1140</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nn_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>no_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>pl_PL</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Polish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>pt_BR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-275</td>
<td>Portuguese</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>pt_PT</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1140</td>
<td>Portuguese</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ro_RO</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ru_RU</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
</tbody>
</table>
**Default Settings for ENCODING, PAPERSIZE, DFLANG, DATESTYLE Options**

<table>
<thead>
<tr>
<th>POSIX Locale</th>
<th>Windows Encoding</th>
<th>UNIX Encoding</th>
<th>z/OS Encoding</th>
<th>DFLANG=</th>
<th>DATESTYLE=</th>
<th>PAPERSIZE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>ru_UA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1154</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_BA</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_ME</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_RS</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sk_SK</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sl_SI</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Slovenian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sq_AL</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-1153</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sr_BA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sr_ME</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sr_RS</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sv_SE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1143</td>
<td>Swedish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ta_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>te_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>th_TH</td>
<td>pc LATIN1</td>
<td>thai</td>
<td>open_ed-1160</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>tr_TR</td>
<td>wturkish</td>
<td>latin5</td>
<td>open_ed-1026</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>uk_UA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>vi_VN</td>
<td>wvietnamese</td>
<td>latin1</td>
<td>open_ed-1164</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_CN</td>
<td>euc-cn</td>
<td>euc-cn</td>
<td>ibm-935</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_HK</td>
<td>ms-950</td>
<td>euc-tw,</td>
<td>ibm-937</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_MO</td>
<td>ms-950</td>
<td>euc-tw,</td>
<td>ibm-937</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_SG</td>
<td>euc-cn</td>
<td>euc-cn</td>
<td>ibm-935</td>
<td>Locale</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_TW</td>
<td>ms-950</td>
<td>euc-tw,</td>
<td>ibm-937</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
</tbody>
</table>

* depends on the platform
Overview to System Options Used in a SAS Session for DBCS

You can use the DBCSLANG= and DBCSTYPE= system options to specify the DBCS encoding values for a SAS session. However, the ENCODING= system option is the recommended method in setting a SAS session for DBCS. The ENCODING= system option has priority. For more information, see "Setting the Encoding of a SAS Session" on page 23. Please see the “ENCODING System Option: UNIX, Windows, and z/OS” on page 587 for more information.

DBCS Values for a SAS Session

The following table shows the supported values for the DBCSLANG= and DBCSTYPE= system options under the z/OS, UNIX, and Windows operating environments.

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks.

<table>
<thead>
<tr>
<th>DBCSLANG=</th>
<th>z/OS DBCSTYPE=</th>
<th>UNIX DBCSTYPE=</th>
<th>Windows DBCSTYPE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>euc</td>
<td>not applicable</td>
</tr>
<tr>
<td>DBCSLANG=</td>
<td>z/OS DBCSTYPE=</td>
<td>UNIX DBCSTYPE=</td>
<td>Windows DBCSTYPE=</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Japanese</td>
<td>pcibm</td>
<td>pcibm</td>
<td>pcibm</td>
</tr>
<tr>
<td>Japanese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>not applicable</td>
<td>sjis</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>ibm</td>
<td>pcibm</td>
<td>pcms</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>dec</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>euc</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>pcibm</td>
<td>pcibm</td>
<td>pcibm</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
</tbody>
</table>
Overview to SAS Language Elements That Use Encoding Values

Overview to SAS Language Elements That Use Encoding Values

When the encoding of the SAS session is different from the encoding of the SAS file or from the data that resides in the SAS file, transcoding must occur. Consider a SAS file that was created in the Western Latin1 encoding, then moved to an IBM mainframe that uses the German EBCDIC encoding. In order for the IBM mainframe to successfully access the file, the SAS data file must be transcoded from the Western Latin1 encoding to the German EBCDIC encoding. For information about transcoding concepts, including SAS language elements that contain options for transcoding, see Chapter 4, “Transcoding for NLS,” on page 27.

SBCS, DBCS, and Unicode Encoding Values for Transcoding Data

The following table presents a list of SBCS, DBCS, and Unicode encoding values for transcoding data for all operating environments. The encoding values are valid for SAS language elements that contain options for transcoding.

*Note:* If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks.

<table>
<thead>
<tr>
<th>Encoding Name</th>
<th>Short Name</th>
<th>Description</th>
<th>Maximum bytes per Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>aarabic</td>
<td>aara</td>
<td>Arabic Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>acroatian</td>
<td>acro</td>
<td>MacOS 36 Croatian</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>acyrillic</td>
<td>acyr</td>
<td>MacOS 7 Cyrillic</td>
<td>1</td>
</tr>
<tr>
<td>agreeek</td>
<td>agrk</td>
<td>Greek Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>ahebrew</td>
<td>aheb</td>
<td>Hebrew Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aiceland</td>
<td>aice</td>
<td>Icelandic Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>any</td>
<td>anye</td>
<td>no transcoding is specified</td>
<td>1</td>
</tr>
<tr>
<td>arabic</td>
<td>arab</td>
<td>Arabic ISO</td>
<td>1</td>
</tr>
<tr>
<td>aroman</td>
<td>arom</td>
<td>Roman Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aromania</td>
<td>arma</td>
<td>MacOS 38 Romania</td>
<td>1</td>
</tr>
<tr>
<td>athai</td>
<td>atha</td>
<td>MacOS 21-Thai</td>
<td>1</td>
</tr>
<tr>
<td>aturkish</td>
<td>atur</td>
<td>Turkish Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>aukrainian</td>
<td>aukr</td>
<td>Ukrainian Macintosh</td>
<td>1</td>
</tr>
<tr>
<td>big5</td>
<td>big5</td>
<td>Traditional Chinese Big5</td>
<td>2</td>
</tr>
<tr>
<td>cyrillic</td>
<td>cyrl</td>
<td>Cyrillic ISO</td>
<td>1</td>
</tr>
<tr>
<td>dec-cn</td>
<td>zvms</td>
<td>Simplified Chinese DEC</td>
<td>4</td>
</tr>
<tr>
<td>dec-jp</td>
<td>jvms</td>
<td>Japanese DEC</td>
<td>2</td>
</tr>
<tr>
<td>dec-tw</td>
<td>yvms</td>
<td>Traditional Chinese DEC</td>
<td>4</td>
</tr>
<tr>
<td>ebcodie037</td>
<td>e037</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcodie275</td>
<td>e275</td>
<td>Brazil EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcodie424</td>
<td>e424</td>
<td>Hebrew EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcodie425</td>
<td>e425</td>
<td>Arabic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcodie500</td>
<td>e500</td>
<td>International EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcodie838</td>
<td>e838</td>
<td>Thai EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>ebcdic870</td>
<td>e870</td>
<td>Central European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic875</td>
<td>e875</td>
<td>Greek EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic905</td>
<td>e905</td>
<td>Latin 3 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic924</td>
<td>e924</td>
<td>European EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1025</td>
<td>ecyr</td>
<td>Cyrillic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1026</td>
<td>etur</td>
<td>Turkish EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1047</td>
<td>elat</td>
<td>Western EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1097</td>
<td>e097</td>
<td>Farsi Bilingual EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1112</td>
<td>ebal</td>
<td>Baltic EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1122</td>
<td>eest</td>
<td>Estonian EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1130</td>
<td>evie</td>
<td>Vietnamese EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1137</td>
<td>e137</td>
<td>Devanagari EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1140</td>
<td>e140</td>
<td>North American EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1141</td>
<td>e141</td>
<td>Austria/Germany EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1142</td>
<td>e142</td>
<td>Denmark/Norway EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1143</td>
<td>e143</td>
<td>Finland/Sweden EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1144</td>
<td>e144</td>
<td>Italy EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1145</td>
<td>e145</td>
<td>Spain EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1146</td>
<td>e146</td>
<td>United Kingdom EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1147</td>
<td>e147</td>
<td>France EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcdic1148</td>
<td>e148</td>
<td>International EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>ebc1149</td>
<td>e149</td>
<td>Iceland EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1153</td>
<td>e153</td>
<td>Latin 2 Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1154</td>
<td>e154</td>
<td>Cyrillic Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1155</td>
<td>e155</td>
<td>Turkey Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1156</td>
<td>e156</td>
<td>Baltic Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1157</td>
<td>e157</td>
<td>Estonia Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1158</td>
<td>e158</td>
<td>Cyrillic Ukraine Euro EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1160</td>
<td>e160</td>
<td>cp1160 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebc1164</td>
<td>e164</td>
<td>cp1164 EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>ebcany</td>
<td>eany</td>
<td>enables you to create a data set that is compatible with all EBCDIC encodings</td>
<td>1</td>
</tr>
<tr>
<td>euc-cn</td>
<td>zeuc</td>
<td>Simplified Chinese EUC</td>
<td>2</td>
</tr>
<tr>
<td>euc-jp</td>
<td>jeuc</td>
<td>Japanese EUC</td>
<td>4</td>
</tr>
<tr>
<td>euc-kr</td>
<td>keuc</td>
<td>Korean EUC</td>
<td>4</td>
</tr>
<tr>
<td>euc-tw</td>
<td>yeuc</td>
<td>Traditional Chinese EUC</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-cn</td>
<td>zfuj</td>
<td>Simplified Chinese FACOM</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-jp</td>
<td>jfug</td>
<td>Japanese FACOM</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-ko</td>
<td>kfuj</td>
<td>Korean FACOM</td>
<td>4</td>
</tr>
<tr>
<td>fujitsu-tw</td>
<td>yfuj</td>
<td>Traditional Chinese FACOM</td>
<td>4</td>
</tr>
<tr>
<td>gb18030</td>
<td>gbke</td>
<td>Simplified Chinese GB18030</td>
<td>4</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>--------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>greek</td>
<td>grek</td>
<td>Greek ISO</td>
<td>1</td>
</tr>
<tr>
<td>hebrew</td>
<td>hebr</td>
<td>Hebrew ISO</td>
<td>1</td>
</tr>
<tr>
<td>hitachi-cn</td>
<td>zhit</td>
<td>Simplified Chinese HITAC</td>
<td>6</td>
</tr>
<tr>
<td>hitachi-jp</td>
<td>jhit</td>
<td>Japanese HITAC</td>
<td>6</td>
</tr>
<tr>
<td>hitachi-ko</td>
<td>khit</td>
<td>Korean HITAC</td>
<td>6</td>
</tr>
<tr>
<td>hitachi-tw</td>
<td>yhit</td>
<td>Traditional Chinese HITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-jp</td>
<td>jhts</td>
<td>Japanese XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-ko</td>
<td>khts</td>
<td>Korean XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hitsas-tw</td>
<td>yhts</td>
<td>Traditional Chinese XHITAC</td>
<td>4</td>
</tr>
<tr>
<td>hp15-tw</td>
<td>yhpz</td>
<td>Traditional Chinese HP15</td>
<td>2</td>
</tr>
<tr>
<td>ibm-1381</td>
<td>zpce</td>
<td>Simplified Chinese PCIBM</td>
<td>2</td>
</tr>
<tr>
<td>ibm-930</td>
<td>j930</td>
<td>Japanese katakana</td>
<td>4</td>
</tr>
<tr>
<td>ibm-933</td>
<td>kibm</td>
<td>Korean IBM</td>
<td>4</td>
</tr>
<tr>
<td>ibm-935</td>
<td>zibm</td>
<td>Simplified Chinese IBM</td>
<td>4</td>
</tr>
<tr>
<td>ibm-937</td>
<td>yibm</td>
<td>Traditional Chinese IBM</td>
<td>4</td>
</tr>
<tr>
<td>ibm-939</td>
<td>jibm</td>
<td>Japanese IBM</td>
<td>4</td>
</tr>
<tr>
<td>ibm-942</td>
<td>j942</td>
<td>Japanese PCIBM</td>
<td>2</td>
</tr>
<tr>
<td>ibm-949</td>
<td>kpce</td>
<td>Korean PCIBM</td>
<td>2</td>
</tr>
<tr>
<td>iso2022cncns</td>
<td>zist</td>
<td>Traditional Chinese ISO-2022</td>
<td>4</td>
</tr>
<tr>
<td>iso2022cngb</td>
<td>ziso</td>
<td>Simplified Chinese ISO-2022</td>
<td>4</td>
</tr>
<tr>
<td>iso2022jp</td>
<td>jiso</td>
<td>Japanese ISO-2022</td>
<td>8</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
<td>------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>iso2022kr</td>
<td>kiso</td>
<td>Korean ISO-2022</td>
<td>4</td>
</tr>
<tr>
<td>kz1048</td>
<td>kz48</td>
<td>Kazakh cp1048 Cyrillic</td>
<td>1</td>
</tr>
<tr>
<td>latin1</td>
<td>lat1</td>
<td>Western ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin2</td>
<td>lat2</td>
<td>Central European ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin3</td>
<td>lat3</td>
<td>Latin 3 ISO 8859/3</td>
<td>1</td>
</tr>
<tr>
<td>latin4</td>
<td>lat4</td>
<td>Latin 4 ISO 8859/4</td>
<td>1</td>
</tr>
<tr>
<td>latin5</td>
<td>lat5</td>
<td>Turkish ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin6</td>
<td>lat6</td>
<td>Baltic ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin7</td>
<td>lat7</td>
<td>Baltic Rim</td>
<td>1</td>
</tr>
<tr>
<td>latin8</td>
<td>lat8</td>
<td>Latin 8 ISO 8859/14</td>
<td>1</td>
</tr>
<tr>
<td>latin9</td>
<td>lat9</td>
<td>European ISO</td>
<td>1</td>
</tr>
<tr>
<td>latin10</td>
<td>lat10</td>
<td>South-eastern Europe</td>
<td>1</td>
</tr>
<tr>
<td>macos-1</td>
<td>jmac</td>
<td>Japanese PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>macos-2</td>
<td>ymac</td>
<td>Traditional Chinese PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>macos-3</td>
<td>kmac</td>
<td>Korean PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>macos-25</td>
<td>zmac</td>
<td>Simplified Chinese PCMAC</td>
<td>2</td>
</tr>
<tr>
<td>ms-932</td>
<td>j932</td>
<td>Japanese PCMS</td>
<td>2</td>
</tr>
<tr>
<td>ms-936</td>
<td>zwin</td>
<td>Simplified Chinese PCMS</td>
<td>2</td>
</tr>
<tr>
<td>ms-949</td>
<td>kwin</td>
<td>Korean PCMS</td>
<td>2</td>
</tr>
<tr>
<td>ms-950</td>
<td>ywin</td>
<td>Traditional Chinese PCMS</td>
<td>2</td>
</tr>
<tr>
<td>msdos720</td>
<td>p720</td>
<td>Arabic MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>msdos737</td>
<td>p737</td>
<td>Greek MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>---------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>msdos775</td>
<td>p775</td>
<td>Baltic MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-037</td>
<td>eous</td>
<td>USA Open Edition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-275</td>
<td>eobr</td>
<td>Brazil OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-424</td>
<td>eoiw</td>
<td>Hebrew OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-425</td>
<td>eoa2</td>
<td>Arabic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-838</td>
<td>eoth</td>
<td>Thai OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-870</td>
<td>eol2</td>
<td>Central European OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-875</td>
<td>eool</td>
<td>Greek OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-905</td>
<td>eol3</td>
<td>Latin 3 Open Edition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-924</td>
<td>eolt</td>
<td>European OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-930</td>
<td>oe30</td>
<td>Katakana OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-933</td>
<td>oe33</td>
<td>Korean OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-935</td>
<td>oe35</td>
<td>Simplified Chinese OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-937</td>
<td>oe37</td>
<td>Traditional Chinese OpenEdition</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-939</td>
<td>oe39</td>
<td>Japanese IBM</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-939E</td>
<td>oe9e</td>
<td>Japanese IBM Extended’</td>
<td>4</td>
</tr>
<tr>
<td>open_ed-1025</td>
<td>eocy</td>
<td>Cyrillic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1026</td>
<td>eotr</td>
<td>Turkish OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1047</td>
<td>eoll</td>
<td>Western OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1097</td>
<td>eofa</td>
<td>Farsi Bilingual OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1112</td>
<td>eobl</td>
<td>Baltic OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>open_ed-1122</td>
<td>eoet</td>
<td>Estonian OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1130</td>
<td>eovi</td>
<td>Vietnamese OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1140</td>
<td>eo40</td>
<td>North American OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1141</td>
<td>eo41</td>
<td>Austria/Germany OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1142</td>
<td>eo42</td>
<td>Denmark/Norway OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1143</td>
<td>eo43</td>
<td>Finland/Sweden OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1144</td>
<td>eo44</td>
<td>Italy OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1145</td>
<td>eo45</td>
<td>Spain OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1146</td>
<td>eo46</td>
<td>United Kingdom OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1147</td>
<td>eo47</td>
<td>France OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1148</td>
<td>eo48</td>
<td>International OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1149</td>
<td>eoIS</td>
<td>Iceland OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1153</td>
<td>eo53</td>
<td>Latin 2 Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1154</td>
<td>eo54</td>
<td>Cyrillic Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1155</td>
<td>eo55</td>
<td>Turkey Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1156</td>
<td>eo56</td>
<td>Baltic Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1157</td>
<td>eo57</td>
<td>Estonia Euro OpenEdition</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>open_ed-1158</td>
<td>eo58</td>
<td>Cyrillic Ukraine Euro 1158 OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1160</td>
<td>eo60</td>
<td>1160 OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1164</td>
<td>eo64</td>
<td>1164 OpenEdition EBCDIC</td>
<td>1</td>
</tr>
<tr>
<td>open_ed-1166</td>
<td>eo66</td>
<td>1166 OpenEdition kz-cyr</td>
<td>1</td>
</tr>
<tr>
<td>pciscii806</td>
<td>p806</td>
<td>Indian PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem437</td>
<td>p437</td>
<td>USA IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem850</td>
<td>p850</td>
<td>Western IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem852</td>
<td>p852</td>
<td>Central European IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem857</td>
<td>p857</td>
<td>Turkish IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem858</td>
<td>p858</td>
<td>European IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem860</td>
<td>p860</td>
<td>Portuguese MS-DOS</td>
<td>1</td>
</tr>
<tr>
<td>pcoem862</td>
<td>p862</td>
<td>Hebrew IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem863</td>
<td>p863</td>
<td>French Canadian IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem864</td>
<td>p864</td>
<td>Arabic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem865</td>
<td>p865</td>
<td>Nordic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem866</td>
<td>p866</td>
<td>Cyrillic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem869</td>
<td>p869</td>
<td>Greek IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem874</td>
<td>p874</td>
<td>Thai IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem921</td>
<td>p921</td>
<td>Baltic IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem922</td>
<td>p922</td>
<td>Estonia IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>pcoem1129</td>
<td>pvie</td>
<td>Vietnamese IBM-PC</td>
<td>1</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
<td>Maximum bytes per Character</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>pc1098</td>
<td>po98</td>
<td>Farsi PC</td>
<td>1</td>
</tr>
<tr>
<td>roman8</td>
<td>rom8</td>
<td>HP Roman 8</td>
<td>1</td>
</tr>
<tr>
<td>shift-jis</td>
<td>sjis</td>
<td>Japanese SJIS</td>
<td>2</td>
</tr>
<tr>
<td>shift-jis04</td>
<td>sjis4</td>
<td>Shift_JIS-2004 surrogate pair support</td>
<td>2</td>
</tr>
<tr>
<td>thai</td>
<td>thai</td>
<td>Thai ISO</td>
<td>1</td>
</tr>
<tr>
<td>us-ascii</td>
<td>ansi</td>
<td>enables you to create a data set that is compatible with all ASCII encodings</td>
<td>1</td>
</tr>
<tr>
<td>utf-8</td>
<td>utf8</td>
<td>Unicode (UTF-8)</td>
<td>4</td>
</tr>
<tr>
<td>utf-16be</td>
<td>u16b</td>
<td>Unicode (UTF-16BE)</td>
<td>2*</td>
</tr>
<tr>
<td>utf-16le</td>
<td>u16l</td>
<td>Unicode (UTF-16LE)</td>
<td>2*</td>
</tr>
<tr>
<td>utf-32be</td>
<td>u32b</td>
<td>Unicode (UTF-32BE)</td>
<td>4**</td>
</tr>
<tr>
<td>utf-32le</td>
<td>u32l</td>
<td>Unicode (UTF-32LE)</td>
<td>4**</td>
</tr>
<tr>
<td>warabic</td>
<td>wara</td>
<td>Arabic Windows</td>
<td>1</td>
</tr>
<tr>
<td>wbaltic</td>
<td>wbal</td>
<td>Baltic Windows</td>
<td>1</td>
</tr>
<tr>
<td>wcyrillic</td>
<td>wcyr</td>
<td>Cyrillic Windows</td>
<td>1</td>
</tr>
<tr>
<td>wgreek</td>
<td>wgrk</td>
<td>Greek Windows</td>
<td>1</td>
</tr>
<tr>
<td>whebrew</td>
<td>wheb</td>
<td>Hebrew Windows</td>
<td>1</td>
</tr>
<tr>
<td>wlatin1</td>
<td>wlt1</td>
<td>Western Windows</td>
<td>1</td>
</tr>
<tr>
<td>wlatin2</td>
<td>wlt2</td>
<td>Central European Windows</td>
<td>1</td>
</tr>
<tr>
<td>wturkish</td>
<td>wtur</td>
<td>Turkish Windows</td>
<td>1</td>
</tr>
<tr>
<td>wvietnamese</td>
<td>wvie</td>
<td>Vietnamese Windows</td>
<td>1</td>
</tr>
</tbody>
</table>

* UTF-16BE and UTF-16LE have a fixed length of two bytes per character.
** UTF-32BE and UTF-32LE have a fixed length of four bytes per character.
Chapter 23
Encoding Values for a SAS Session

UNIX Encoding Values

The encodings in the following tables are valid in UNIX environments.

*Note:* If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks.

**Table 23.1 Single-Byte Encodings for UNIX**

<table>
<thead>
<tr>
<th>ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arabic</td>
<td>Arabic (ISO 8859-6)</td>
</tr>
<tr>
<td>cyrillic</td>
<td>Cyrillic (ISO 8859-5)</td>
</tr>
<tr>
<td>greek</td>
<td>Greek (ISO 8859-7)</td>
</tr>
<tr>
<td>hebrew</td>
<td>Hebrew (ISO 8859-8)</td>
</tr>
<tr>
<td>latin1</td>
<td>Western (ISO 8859-1)</td>
</tr>
<tr>
<td>latin2</td>
<td>Central Europe (ISO 8859-2)</td>
</tr>
<tr>
<td>latin5</td>
<td>Turkish (ISO 8859-9)</td>
</tr>
<tr>
<td>latin6</td>
<td>Baltic (ISO 8859-4)</td>
</tr>
<tr>
<td>Latin7</td>
<td>Baltic (ISO 8859-13)</td>
</tr>
<tr>
<td>latin8</td>
<td>Celtic (ISO 8859-14)</td>
</tr>
</tbody>
</table>
Table 23.2  Double-Byte Encodings for UNIX

<table>
<thead>
<tr>
<th>ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>latin9</td>
<td>European (ISO 8859-15)</td>
</tr>
<tr>
<td>latin10</td>
<td>South-Eastern European (ISO 8859-16)</td>
</tr>
<tr>
<td>thai</td>
<td>Thai (ISO 8859-11)</td>
</tr>
</tbody>
</table>

UNIX also supports the utf-8 Unicode encoding.

Windows Encoding Values

The encodings in the following tables are valid in the Windows operating environment.

Note: If an encoding-value contains a hyphen (-), enclose the encoding value in quotation marks.

Table 23.3  Single-Byte Encodings for Windows

<table>
<thead>
<tr>
<th>Description</th>
<th>Windows ENCODING= Value</th>
<th>MS-DOS ENCODING= Value</th>
<th>IBM-PC ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>warabic</td>
<td>msdos720</td>
<td>pcoem864</td>
</tr>
<tr>
<td>Baltic</td>
<td>wbaltic</td>
<td>msdos775</td>
<td>pcoem921</td>
</tr>
<tr>
<td>Central Europe</td>
<td>wlatin2</td>
<td>not applicable</td>
<td>pcoem852</td>
</tr>
<tr>
<td>Cyrillic</td>
<td>wcyrillic</td>
<td>not applicable</td>
<td>pcoem866 pcoem855</td>
</tr>
<tr>
<td>Estonia</td>
<td>wbaltic</td>
<td>not applicable</td>
<td>pcoem922</td>
</tr>
<tr>
<td>Description</td>
<td>Windows ENCODING= Value</td>
<td>MS-DOS ENCODING= Value</td>
<td>IBM-PC ENCODING= Value</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>European</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem858</td>
</tr>
<tr>
<td>Farsi</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pc1098</td>
</tr>
<tr>
<td>French Canadian</td>
<td>wlatin1</td>
<td>not applicable</td>
<td>pcoem863</td>
</tr>
<tr>
<td>Greek</td>
<td>wgreek</td>
<td>msdos737</td>
<td>not applicable</td>
</tr>
<tr>
<td>Hebrew</td>
<td>whebrew</td>
<td>not applicable</td>
<td>pcoem862</td>
</tr>
<tr>
<td>Indian Script Code</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pciscii806</td>
</tr>
<tr>
<td>Nordic</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem865</td>
</tr>
<tr>
<td>Portuguese</td>
<td>wlatin1</td>
<td>pcoem860</td>
<td>not applicable</td>
</tr>
<tr>
<td>Thai</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem874</td>
</tr>
<tr>
<td>Turkish</td>
<td>wturkish</td>
<td>not applicable</td>
<td>pcoem857</td>
</tr>
<tr>
<td>USA</td>
<td>wlatin1</td>
<td>not applicable</td>
<td>pcoem437</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>wvietnamese</td>
<td>not applicable</td>
<td>not applicable</td>
</tr>
<tr>
<td>Western</td>
<td>wlatin1</td>
<td>not applicable</td>
<td>pcoem858</td>
</tr>
</tbody>
</table>

Table 23.4 Windows Double-Byte Encodings

<table>
<thead>
<tr>
<th>Description</th>
<th>PCMS ENCODING= Value</th>
<th>No Vendor ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional Chinese</td>
<td>ms-950</td>
<td>big5</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>ms-936</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>ms-932</td>
<td>shift-jis</td>
</tr>
<tr>
<td>Korean</td>
<td>ms-949</td>
<td>not applicable</td>
</tr>
</tbody>
</table>

Note: Windows also supports the utf-8 Unicode encoding.
Note: If an encoding-value contains a hyphen (-), enclose the encoding value in quotation marks.

Table 23.5 Single-Byte Encodings for z/OS

<table>
<thead>
<tr>
<th>Encoding ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC037</td>
<td>EBCDIC cp037- Old North America</td>
</tr>
<tr>
<td>EBCDIC275</td>
<td>EBCDIC cp275-Brazil</td>
</tr>
<tr>
<td>EBCDIC425</td>
<td>EBCDIC cp425-Arabic</td>
</tr>
<tr>
<td>EBCDIC838</td>
<td>EBCDIC cp838-Thai</td>
</tr>
<tr>
<td>EBCDIC870</td>
<td>EBCDIC cp870-Central Europe</td>
</tr>
<tr>
<td>EBCDIC875</td>
<td>EBCDIC cp875-Greek</td>
</tr>
<tr>
<td>EBCDIC905</td>
<td>EBCDIC cp905-Latin 3</td>
</tr>
<tr>
<td>EBCDIC924</td>
<td>EBCDIC cp924-Western Europe</td>
</tr>
<tr>
<td>EBCDIC1025</td>
<td>EBCDIC cp1025-Cyrillic</td>
</tr>
<tr>
<td>EBCDIC1026</td>
<td>EBCDIC cp1026-Turkish</td>
</tr>
<tr>
<td>EBCDIC1047</td>
<td>EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>EBCDIC1097</td>
<td>EBCDIC cp1097-Farsi Bilingual</td>
</tr>
<tr>
<td>EBCDIC1112</td>
<td>EBCDIC cp1112-Baltic</td>
</tr>
<tr>
<td>EBCDIC1122</td>
<td>EBCDIC cp1122-Estonian</td>
</tr>
<tr>
<td>EBCDIC1130</td>
<td>EBCDIC cp1130-Vietnamese</td>
</tr>
<tr>
<td>EBCDIC1137</td>
<td>EBCDIC cp1137-Devanagari</td>
</tr>
<tr>
<td>EBCDIC1140</td>
<td>EBCDIC cp1140-North America</td>
</tr>
<tr>
<td>EBCDIC1141</td>
<td>EBCDIC cp1141-German/Austrian</td>
</tr>
<tr>
<td>EBCDIC1142</td>
<td>EBCDIC cp1142-Danish/Norwegian</td>
</tr>
<tr>
<td>EBCDIC1143</td>
<td>EBCDIC cp1143-Finnish/Swedish</td>
</tr>
<tr>
<td>EBCDIC1144</td>
<td>EBCDIC cp1144-Italian</td>
</tr>
<tr>
<td>EBCDIC1145</td>
<td>EBCDIC cp1145-Spanish</td>
</tr>
<tr>
<td>Encoding ENCODING= Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>EBCDIC1146</td>
<td>EBCDIC cp1146-English (UK)</td>
</tr>
<tr>
<td>EBCDIC1147</td>
<td>EBCDIC cp1147-French</td>
</tr>
<tr>
<td>EBCDIC1148</td>
<td>EBCDIC cp1148-International</td>
</tr>
<tr>
<td>EBCDIC1149</td>
<td>EBCDIC cp1149-Iceland</td>
</tr>
<tr>
<td>EBCDIC1153</td>
<td>EBCDIC cp1153-Latin 2 Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1154</td>
<td>EBCDIC cp1154-Cyrillic Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1155</td>
<td>EBCDIC cp1155-Turkey with euro</td>
</tr>
<tr>
<td>EBCDIC1156</td>
<td>EBCDIC cp1156-Baltic Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1157</td>
<td>EBCDIC cp1157-Estonia with euro</td>
</tr>
<tr>
<td>EBCDIC1158</td>
<td>EBCDIC cp1158-Cyrillic Ukraine with euro</td>
</tr>
<tr>
<td>OPEN_ED-037</td>
<td>OpenEdition EBCDIC cp037-Old North America</td>
</tr>
<tr>
<td>OPEN_ED-275</td>
<td>OpenEdition EBCDIC cp275-Brazil</td>
</tr>
<tr>
<td>OPEN_ED-425</td>
<td>OpenEdition EBCDIC cp425-Arabic</td>
</tr>
<tr>
<td>OPEN_ED-838</td>
<td>OpenEdition EBCDIC cp838-Thai</td>
</tr>
<tr>
<td>OPEN_ED-870</td>
<td>OpenEdition EBCDIC cp870-Central Europe</td>
</tr>
<tr>
<td>OPEN_ED-875</td>
<td>OpenEdition EBCDIC cp875-Greek</td>
</tr>
<tr>
<td>OPEN_ED-905</td>
<td>OpenEdition EBCDIC cp905-Latin 3</td>
</tr>
<tr>
<td>OPEN_ED-924</td>
<td>OpenEdition EBCDIC cp924-Western Europe</td>
</tr>
<tr>
<td>OPEN_ED-1025</td>
<td>OpenEdition EBCDIC cp1025-Cyrillic</td>
</tr>
<tr>
<td>OPEN_ED-1026</td>
<td>OpenEdition EBCDIC cp1026-Turkish</td>
</tr>
<tr>
<td>OPEN_ED-1047</td>
<td>OpenEdition EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>OPEN_ED_1097</td>
<td>OpenEdition EBCDIC cp1097-Farsi Bilingual</td>
</tr>
<tr>
<td>OPEN_ED-1112</td>
<td>OpenEdition EBCDIC cp1112-Baltic</td>
</tr>
<tr>
<td>OPEN_ED-1122</td>
<td>OpenEdition EBCDIC cp1122-Estonian</td>
</tr>
<tr>
<td>OPEN_ED-1130</td>
<td>OpenEdition EBCDIC cp1130-Vietnamese</td>
</tr>
</tbody>
</table>
### Table 23.6  Double-Byte Encodings for z/OS

<table>
<thead>
<tr>
<th>Description</th>
<th>ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>OPEN_ED-939</td>
</tr>
<tr>
<td>Korean</td>
<td>OPEN_ED-933</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>OPEN_ED-935</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>OPEN_ED-937</td>
</tr>
</tbody>
</table>
Part 12

Appendixes

Appendix 1
Additional NLS Language Elements ........................................ 731

Appendix 2
Time Zone IDs and Time Zone Names ................................. 819
Appendix 1

Additional NLS Language Elements

Additional NLS Language Elements .............................................. 732

Dictionary .................................................................................. 732
EURDFDDw. Format ................................................................. 732
EURDFDEw. Format ................................................................. 734
EURDFDNw. Format ................................................................. 735
EURDFDTw.d Format ............................................................... 736
EURDFDWNw. Format ............................................................. 738
EURDFMNw. Format ................................................................. 741
EURDFMYw. Format ................................................................. 742
EURDFWDXw. Format ............................................................... 744
EURDFWKXw. Format ............................................................... 746
EURFRATSw.d Format ............................................................... 749
EURFRBEFw.d Format ............................................................... 750
EURFRCHFw.d Format ............................................................... 752
EURFRDEMw.d Format ............................................................... 753
EURFRDKKw.d Format ............................................................... 754
EURFRESwp.d Format ............................................................... 756
EURFRFIMw.d Format ............................................................... 757
EURFRFRFw.d Format ............................................................... 758
EURFRGBPw.d Format ............................................................... 759
EURFRGRDw.d Format ............................................................... 761
EURFRHUFw.d Format ............................................................... 762
EURFRHUFw.d Format ............................................................... 763
EURFRITLw.d Format ............................................................... 765
EURFRITLw.d Format ............................................................... 766
EURFRNLGw.d Format ............................................................... 767
EURFRNOKw.d Format ............................................................... 769
EURFRPLZw.d Format ............................................................... 770
EURFRPTEw.d Format ............................................................... 771
EURFRROLLw.d Format ............................................................. 773
EURFRRURw.d Format ............................................................... 774
EURFRSEKw.d Format ............................................................... 775
EURFRSITw.d Format ............................................................... 777
EURFRTRLw.d Format ............................................................... 778
EURTOATSw.d Format ............................................................... 779
EURTOBEFw.d Format ............................................................... 780
EURTOCHFw.d Format ............................................................... 782
EURTOCZKw.d Format ............................................................... 783
EURTODEMw.d Format ............................................................... 784
EURTODKKw.d Format ............................................................... 786
EURTOESPw.d Format ............................................................... 787
### Additional NLS Language Elements

The following EUR language elements have been replaced with NL language elements. The EUR elements are supported in SAS 9.3, but SAS recommends that you use the NL elements.

#### Dictionary

**EURDFDDw. Format**

Writes international date values in the form `dd.mm.yy` or `dd.mm.yyyy`.

- **Category:** Date and Time
- **Alignment:** Right

**Syntax**

```r
EURDFDDw;
```

**Syntax Description**

`w` specifies the width of the output field.

**Default** 8 (except Finnish, which is 10)
Tip
When \( w \) is from 2 to 5, SAS prints as much of the month and day as possible. When \( w \) is 7, the date appears as a two-digit year without slashes, and the value is right-aligned in the output field.

Details
The EURDFDD\( w \). format writes SAS date values in the form \( dd.mm.yy \) or \( dd.mm.yyyy \), where

- \( dd \) is the two-digit integer that represents the day of the month.
- \( mm \) is the two-digit integer that represents the month.
- \( yy \) or \( yyyy \) is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= system option.

Example
The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Spanish.

```sas
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the international date value. The third PUT statement uses the French language prefix in the format to write the international date value. Therefore, the value of the DFLANG= option is ignored.

```sas
options dflang=spanish;
data _null_;  
  input day;  
  put day eurdfdd8.;  
data lines;  
  15342  

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfdd8.;</td>
<td>02.01.02</td>
</tr>
<tr>
<td>put date espdfdd8.;</td>
<td>02.01.02</td>
</tr>
<tr>
<td>put date fradfdd8.;</td>
<td>02/01/02</td>
</tr>
</tbody>
</table>
**EURDFDEw. Format**

Writes international date values in the form `ddmmmyy` or `ddmmmyyyy`.

**Category:** Date and Time  
**Alignment:** Right

**Syntax**

`EURDFDEw;`

**Syntax Description**

`w`  
specifies the width of the output field.

**Default**  7 (except Finnish)  
**Range**  5–9 (except Finnish)  
**Note**  If you use the Finnish (FIN) language prefix, the `w` range is 9–10 and the default is 9.

**Details**

The `EURDFDEw` format writes SAS date values in the form `ddmmmyy` or `ddmmmyyyy`:

- `dd` is an integer that represents the day of the month.
- `mmm` is the first three letters of the month name.
- `yy` or `yyyy` is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the `DFLANG=` system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “`DFLANG=` System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the `DFLANG=` option.

**Note:** The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set, such as UTF-8.

**Example**

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the `DFLANG=` system option is set to Spanish.
options dflang=spanish;

The second PUT statement uses the Spanish language prefix in the format to write the international date value in Spanish. The third PUT statement uses the French language prefix in the format to write the international date value in French. Therefore, the value of the DFLANG= option is ignored.

options dflang=spanish;
data _null_;  
  input day;  
  put day eurdfde9.;  
  put day espdfde9.;  
  put day fradfde9.;  
datalines;  
  15342  
;  

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfde9.;</td>
<td>02ene2002</td>
</tr>
<tr>
<td>put date espdfde9.;</td>
<td>02ene2002</td>
</tr>
<tr>
<td>put date fradfde9.;</td>
<td>02jan2002</td>
</tr>
</tbody>
</table>

**EURDFDNw. Format**

Writes international date values as the day of the week.

- **Category:** Date and Time
- **Alignment:** Right

**Syntax**

EURDFDNw.

**Syntax Description**

w

specifies the width of the output field.

- **Default:** 1
- **Range:** 1–32

**Details**

The EURDFDNw. format writes SAS date values in the form *day-of-the-week*:
day-of-the-week
is represented as 1=Monday, 2=Tuesday, and so on.

You can set the language for the SAS session with the DFLANG= system option.
(Because the SAS Installation Representative usually sets a default language for the site,
you might be able to skip this step.) If you work with dates in multiple languages, you
can replace the EUR prefix with a language prefix. See “DFLANG= System Option:
UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you
specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some
formats work correctly using non-European encodings. When running in a DBCS
environment, the default format width and max width are larger than in the single-
byte system to allow formats to use a double-byte representation of certain
characters. However, you must use a session encoding that supports the European
characters set like UTF-8.

Example

The example table uses the input value 15342, which is the SAS date value that
corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG=
system option is set to Spanish.

options dflang=spanish;

The second PUT statement uses the Spanish language prefix in the format to write the
day of the week in Spanish. The third PUT statement uses the Italian language prefix in
the format to write the day of the week in Italian. Therefore, the value of the DFLANG=
option is ignored.

options dflang=spanish;
data _null_;  
  input day;  
  put day eurdfdn.;  
  put day espdfdn.;  
  put day itadfdn.;  
datalines;  
  15342  
;  

tables

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---+-----</td>
</tr>
<tr>
<td>put day eurdfdn.;</td>
<td>3</td>
</tr>
<tr>
<td>put day espdfdn.;</td>
<td>3</td>
</tr>
<tr>
<td>put day itadfdn.;</td>
<td>3</td>
</tr>
</tbody>
</table>

EURDFDTw.d Format

Writes international datetime values in the form ddmmyy:hh:mm:ss.ss or ddmmyyyy hh:mm:ss.ss.
Syntax

EURDFDT\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 16

Range 7–40

Tip If you want to write a SAS datetime value with the date, hour, and seconds, the width (\textit{w}) must be at least 16. Add an additional two places to the width if you want to return values with optional decimal fractions of seconds.

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Range 1–39

Restrictions must be less than \textit{w}

If \textit{w} – \textit{d} < 17, SAS truncates the decimal values.

Details

The EURDFDT\textit{w.d} format writes SAS datetime values in the form \textit{ddmmmyy:hh:mm:ss.ss}:

\textit{dd}

is an integer that represents the day of the month.

\textit{mmm}

is the first three letters of the month name.

\textit{yy} or \textit{yyyy}

is a two-digit or four-digit integer that represents the year.

\textit{hh}

is the number of hours that range from 00 through 23.

\textit{mm}

is the number of minutes that range from 00 through 59.

\textit{ss.ss}

is the number of seconds that range from 00 through 59 with the fraction of a second following the decimal point.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.
Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Example

The example table uses the input value of 1347453583, which is the SAS datetime value that corresponds to September 12, 2002, at 12:39:43 p.m. The first PUT statement assumes that the DFLANG= system option is set to German.

options dflang=german;

The second PUT statement uses the German language prefix in the format to write the international datetime value in German. The third PUT statement uses the Italian language prefix in the format to write the international datetime value in Italian. The value of the DFLANG= option, therefore, is ignored.

options dflang=german;
data _null_;  
input date;  
put date= ;  
put date eurdfdt20.;  
put date deudfdt20.;  
put date itadfdt20.;  
datalines;  
1347453583;
;
run;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfdt20.;</td>
<td>12Sep2002:12:39:43</td>
</tr>
<tr>
<td>put date deudfdt20.;</td>
<td>12Sep2002:12:39:43</td>
</tr>
<tr>
<td>put date itadfdt20.;</td>
<td>12Set2002:12:39:43</td>
</tr>
</tbody>
</table>

EURDFDWNw. Format

Writes international date values as the name of the day.

Category: Date and Time
Alignment: Right

Syntax

EURDFDWNw.
Syntax Description

\( w \)

specifies the width of the output field.

The default depends on the language prefix that you use. The following table shows the default value for each language:

<table>
<thead>
<tr>
<th>Language</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>9</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>9</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>10</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>7</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>7</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>9</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>11</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>8</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>10</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>9</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>9</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>10</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>7</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>12</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>13</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>11</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>10</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>9</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>7</td>
</tr>
<tr>
<td>Swiss-French (FRS)</td>
<td>8</td>
</tr>
<tr>
<td>Swiss-German (DES)</td>
<td>10</td>
</tr>
</tbody>
</table>
Default depends on the language prefix you use.

Range 1–32

Tip If you omit w, SAS prints the entire name of the day.

Details

If necessary, SAS truncates the name of the day to fit the format width. The EURDFDWNw. format writes SAS date values in the form day-name:

day-name
  is the name of the day.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Example

The following example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes that the DFLANG= system option is set to French.

options dflang=french;
put day eurdfdwn8.;

The second PUT statement uses the French language prefix in the format to write the day of the week in French. The third PUT statement uses the Spanish language prefix in the format to write the day of the week in Spanish. Therefore, the value of the DFLANG= option is ignored.

options dflang=french;
data _null_; input day;
put day eurdfdwn8.;
put day fradfdwn8.;
put day espdfdwn8.;
datalines;
15344
;
run;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1</td>
<td></td>
</tr>
</tbody>
</table>
EURDFMNw. Format

Writes international date values as the name of the month.

**Category:** Date and Time  
**Alignment:** Right

**Syntax**

EURDFMNw.

**Syntax Description**

- \( w \)  
  - Specifies the width of the output field.
  - **Default:** 9 (except for Finnish and Spanish)
  - **Range:** 1–32
  - **Note:** If you use the Finnish (FIN) language prefix, the default value for \( w \) is 11. If you use the Spanish (ESP) language prefix, the default value for \( w \) is 10.

**Details**

If necessary, SAS truncates the name of the month to fit the format width. The EURDFMNw. format writes SAS date values in the form `month-name:

- `month-name`  
  - Is the name of the month.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

**Note:** The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.
Example

The example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes that the DFLANG= system option is set to Italian.

```
options dflang=ita;
```

The second PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. The third PUT statement uses German language prefix in the format to write the name of the month in German. Therefore, the value of the DFLANG= option is ignored.

```
options dflang=ita;
data _null_;  
input day;  
put day eurdfmn10.;  
put day itadfmn10.;  
put day deudfmn10.;  
datalines;  
15344  
;  
run;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfmn10.;</td>
<td>janvier</td>
</tr>
<tr>
<td>put date itadfmn10.;</td>
<td>Gennaio</td>
</tr>
<tr>
<td>put date deudfmn10.;</td>
<td>Januar</td>
</tr>
</tbody>
</table>

**EURDFMYw. Format**

Writes international date values in the form **mmmyy** or **mmmyyyy**.

**Category:** Date and Time

**Alignment:** Right

**Syntax**

```
EURDFMYw:
```

**Syntax Description**

* w

specifies the width of the output field.

**Default** 5 (except for Finnish)
Range  5–7

Note  If you use the Finnish (FIN) language prefix, the value for \( w \) must be 8, which is the default value.

Details

The EURDFMY\( _w \) format writes SAS date values in the form \( mmmyy \), where

\( mmm \)

is the first three letters of the month name.

\( yy \) or \( yyyy \)

is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Example

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Spanish.

```sas
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the name of the month in Spanish. The third PUT statement uses the French language prefix in the format to write the name of the month in French. Therefore, the value of the DFLANG= option is ignored.

```sas
options dflang=spanish;
data _null_;  
input date;  
   put date eurdfmy7.;  
   put date espdfmy7.;  
   put date fradfmy7.;  
datalines;
15342
;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfmy7.;</td>
<td>ene2002</td>
</tr>
</tbody>
</table>
EURDFWDXw. Format

Writes international date values as the name of the month, the day, and the year in the form dd month-name yy (or yyyy).

**Category:** Date and Time

**Alignment:** Right

**Syntax**

EURDFWDXw:

**Syntax Description**

\( w \)

specifies the width of the output field.

The default depends on the language prefix that you use. The following table shows the default value for each language:

<table>
<thead>
<tr>
<th>Language</th>
<th>Maximum</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>40</td>
<td>18</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Language</td>
<td>Maximum</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>37</td>
<td>23</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Swiss-French (FRS)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Swiss-German (DES)</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

- **Default**: depends on the language prefix you use.
- **Range**: 3–(maximum width)
- **Tip**: If the value for \( w \) is too small to include the complete day of the week and the month, SAS abbreviates as necessary.

### Details

The EURDFWDXw. format writes SAS date values in the form `dd month-name yy` or `dd month-name yyyy`:

- `dd` is an integer that represents the day of the month.
- `month-name` is the name of the month.
- `yy` or `yyyy` is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

**Note**: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain
characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Comparisons

The EURDFWKKXw. format is the same as the EURDFWDXw. format except that EURDFWKKX w. format adds the day-of-week in front of dd.

Example

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Dutch.

```sas
options dflang=dutch;
```

The second PUT statement uses the Dutch language prefix in the format to write the name of the month in Dutch. The third PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. Therefore, the value of the DFLANG= option is ignored.

```sas
options dflang=dutch;
data _null_;  
  input date;  
  put date eurdfwdx29.;  
  put date nlddfwdx29.;  
  put date itadfwdx17.;  
  datalines;  
  15342  
;  
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day eurdfwdx29.;</td>
<td>2 januari 2002</td>
</tr>
<tr>
<td>put day nlddfwdx29.;</td>
<td>2 januari 2002</td>
</tr>
<tr>
<td>put day itadfwdx17.;</td>
<td>02 Gennaio 1998</td>
</tr>
</tbody>
</table>

---

EURDFWKKXw. Format

Writes international date values as the name of the day and date in the form day-of-week, dd month-name yy (or yyyy).

| Category: | Date and Time |
| Alignment: | Right |
Syntax

**EURDFWKX**w.

**Syntax Description**

w specifies the width of the output field.

The default depends on the language prefix that you use. The following table shows the default value for each language:

<table>
<thead>
<tr>
<th>Language</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>2</td>
<td>38</td>
<td>28</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>2</td>
<td>40</td>
<td>27</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>3</td>
<td>40</td>
<td>27</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>2</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>2</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>2</td>
<td>38</td>
<td>28</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>2</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>3</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>3</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>3</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>3</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>3</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td>3</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>2</td>
<td>40</td>
<td>34</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>3</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>2</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>3</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>1</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>3</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Swiss-French (FRS)</td>
<td>3</td>
<td>26</td>
<td>26</td>
</tr>
</tbody>
</table>
Details

The EURDFWKXw. format writes SAS date values in the form day-of-week, dd month-name yy (or yyyy):

day-of-week
  is the name of day.

dd
  is an integer that represents the day of the month.

month-name
  is the name of the month.

yy or yyyy
  is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are larger than in the single-byte system to allow formats to use a double-byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8.

Comparisons

The EURDFWKXw. format is the same as the EURDFWDXw. format except that EURDFWKXw. format adds day-of-week in front of dd.

Example

The example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes that the DFLANG= system option is set to German.

options dflang=German;

The second PUT statement uses the German language prefix in the format to write the name of the month in German. The third PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. Therefore, the value of the DFLANG= option is ignored.
options dflang=german;
data _null_;  
input date;
put date eurdfwkx30.;
put date deudfwkx30.;
put date itadfwkx17.;
datalines;
  15344
;  
run;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfwkx30.;</td>
<td>Freitag, 4. Januar 2002</td>
</tr>
<tr>
<td>put date deudfwkx30.;</td>
<td>Freitag, 4. Januar 2002</td>
</tr>
<tr>
<td>put date itadfwkx17.;</td>
<td>Ven, 04 Gen 2002</td>
</tr>
</tbody>
</table>

EURFRATS\textit{w.d} Format

Converts an amount from Austrian schillings to euros.

- **Category:** Currency Conversion
- **Alignment:** Right

### Syntax

\texttt{EURFRATSW.d}

### Syntax Description

\texttt{w}

- specifies the width of the output field.
- **Default:** 6

\texttt{d}

- specifies the number of digits to the right of the decimal point in the numeric value.

### Details

The EURFRATS \textit{w.d} format converts an amount from Austrian schillings to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRATS\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.
Example
The following table shows input values in Austrian schillings, SAS statements, and the conversion results in euros.

```sas
data _null_
  input amount;
  put amount eurfrats5.;
  put amount eurfrats9.2;
  datalines;
50
5234.56
52345;
run;

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrats5.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E3,63</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrats5.;</td>
<td>E380</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E380,41</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrats5.;</td>
<td>3.804</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E3.804,06</td>
</tr>
</tbody>
</table>
```

**EURFRBEFw.d Format**
Converts an amount from Belgian francs to euros.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Currency Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment:</td>
<td>Right</td>
</tr>
</tbody>
</table>

**Syntax**

```sas
EURFRBEFw.d
```

**Syntax Description**

`w`
specifies the width of the output field.
**Default 6**

\[ \textit{d} \]

Specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRBEF\textit{w.d} format converts an amount from Belgian francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRBEF\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Belgian francs, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrbef5.;  
  put amount eurfrbef9.2;  
  datalines;  
  50  
  5234.56  
  52345  
  ;  
run;  
8 put amount eurfrbef5.;  
9 put amount eurfrbef9.2;  
10 datalines;  
  E1  
  E1,24  
  E130  
  E129,76  
  1.298  
  E1,297,60
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrbef5.; E1</td>
<td>E1</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrbef5.; E130</td>
<td>E130</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrbef5.; 1.298</td>
<td>1.298</td>
</tr>
</tbody>
</table>
**EURFRCHFw.d Format**

Converts an amount from Swiss francs to euros.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

\[ \text{EURFRCHFw.d} \]

**Syntax Description**

\( w \)

specifies the width of the output field.

Default \( 6 \)

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRCHFw.d format converts an amount from Swiss francs to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRCHFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Swiss francs, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrchf5.;  
  put amount eurfrchf9.2;  
  datalines;  
50  
1234.56  
12345  
;  
run;  
SAS Log:  
3      put amount eurfrchf5.;  
4      put amount eurfrchf9.2;  
5      datalines;  
E31  
E31,17  
E770  
E769,53  
7.695
```

---

*Appendix 1 • Additional NLS Language Elements*
EURFRDEMw.d Format

Converts an amount from Deutsche marks to euros.

**Category:** Currency Conversion

**Alignment:** Right

### Syntax

\texttt{EURFRDEMw.d}

### Syntax Description

\texttt{w}

specifies the width of the output field.

Default 6

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

### Details

The EURFRDEMw.d format converts an amount from Deutsche marks to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRDEMw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

### Example

The following table shows input values in Deutsche marks, SAS statements, and the conversion results in euros.

```sas
data _null_;

put amount eurfrchf5.;
put amount eurfrchf9.2;

E31
E31,17

put amount eurfrchf5.;
put amount eurfrchf9.2;

E770
E769,53

put amount eurfrchf5.;
put amount eurfrchf9.2;

7.695
E7.694,94
```

---

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrchf5.;</td>
<td>E31</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E31,17</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrchf5.;</td>
<td>E770</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E769,53</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrchf5.;</td>
<td>7.695</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E7.694,94</td>
</tr>
</tbody>
</table>
input amount;
put amount eurfrdem5.;
put amount eurfrdem9.2;
datalines;
50
1234.56
12345
;
run;
8       put amount eurfrdem5.;
9       put amount eurfrdem9.2;
10      datalines;
      E26
      E25,56
      E631
      E631,22
      6.312
      E6.311,90

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrdem5.;</td>
<td>E26</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrdem9.2;</td>
<td>E25,56</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrdem5.;</td>
<td>E631</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrdem9.2;</td>
<td>E631,22</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrdem5.;</td>
<td>6.312</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrdem9.2;</td>
<td>E6.311,90</td>
</tr>
</tbody>
</table>

**EURFRDKKw.d Format**

Converts an amount from Danish kroner to euros.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

EURFRDKKw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

Default: 6
\(d\) specifies the number of digits to the right of the decimal point in the numeric value.

### Details

The EURFRDKKw.d format converts an amount from Danish kroner to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRDKKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

### Example

The following table shows input values in Danish kroner, SAS statements, and the conversion results in euros.

```
data _null_;  input amount;  put amount eurfrdkk5.;  put amount eurfrdkk9.2;  datalines;  50  1234.56  12345  ;  run;  
```

SAS log:

```
3      put amount eurfrdkk5.;  4      put amount eurfrdkk9.2;  5      datalines;  
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrdkk5.;</td>
<td>E7</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrdkk9.2;</td>
<td>E6,68</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrdkk5.;</td>
<td>E165</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrdkk9.2;</td>
<td>E164,83</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrdkk5.;</td>
<td>1.648</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrdkk9.2;</td>
<td>E1.648,18</td>
</tr>
</tbody>
</table>
EURFRESPw.d Format

Converts an amount from Spanish peseta to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRESPw.d

Syntax Description

w

specifies the width of the output field.

Default 6

d

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRESPw.d format converts an amount from Spanish peseta to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRESPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in Spanish peseta, SAS statements, and the conversion results in euros.

data _null_
  input amount;
  put amount eurfresp5.;
  put amount eurfresp9.2;
  datalines;
  200
  20234.56
  202345 ;
run;

8       put amount eurfresp5.;
9       put amount eurfresp9.2;
10      datalines;
E1
E1,20
E122
E121,61
1.216
EURFRFIMw.d Format

Converts an amount from Finnish markkas to euros.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

`EURFRFIMw.d`

**Syntax Description**

`w`

specifies the width of the output field.

Default 6

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRFIMw.d format converts an amount from Finnish markkas to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRFIMw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Finnish markkas, SAS statements, and the conversion results in euros.

```sas
data _null_;  
```

```
<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>put amount eurfrsp5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsp9.2;</td>
<td>E1,20</td>
</tr>
<tr>
<td>20234.56</td>
<td>put amount eurfrsp5.;</td>
<td>E122</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsp9.2;</td>
<td>E121,61</td>
</tr>
<tr>
<td>202345</td>
<td>put amount eurfrsp5.;</td>
<td>1.216</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsp9.2;</td>
<td>E1.216,12</td>
</tr>
</tbody>
</table>
```

input amount;
put amount eurfrfim5.;
put amount eurfrfim9.2;
datalines;
50
1234.56
12345
;
run;
8    put amount eurfrfim5.;
9    put amount eurfrfim9.2;
10   datalines;
   E8
   E8,41
E208
E207,64
2.076
E2.076,28

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrfim5.; E8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfim9.2.; E8,41</td>
<td></td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrfim5.; E208</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfim9.2.; E207,64</td>
<td></td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrfim5.; 2.076</td>
<td></td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfim9.2.; E2.076,28</td>
<td></td>
</tr>
</tbody>
</table>

**EURFRFRFw.d Format**

Converts an amount from French francs to euros.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

EURFRFRFw.d

**Syntax Description**

w

specifies the width of the output field.

Default: 6
specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRFRFw.d format converts an amount from French francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRFRFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in French francs, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrfrf5.;  
  put amount eurfrfrf9.2;  
datalines;  
50  
1234.56  
12345  
;  
run;  

SAS log:  
  E8  
  E7,62  
  E188  
  E188,21  
  1.882  
  E1.881,98
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrfrf5.;</td>
<td>E8</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfrf9.2;</td>
<td>E7,62</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrfrf5.;</td>
<td>E188</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfrf9.2;</td>
<td>E188,21</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrfrf5.;</td>
<td>1.882</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfrf9.2;</td>
<td>E1.881,98</td>
</tr>
</tbody>
</table>

**EURFRGBPw.d Format**

Converts an amount from British pounds to euros.
Syntax
EURFRGBPw.d

Syntax Description

w
specifies the width of the output field.

Default 6

d
specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRGBPw.d format converts an amount from British pounds to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRGBPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in British pounds, SAS statements, and the conversion results in euros.

data _null_
  input amount;
  put amount eurfrgbp5.;
  put amount eurfrgbp9.2;
  datalines;
50
1234.56
12345
;
run;
SAS log:
5  put amount eurfrgbp5.;
4  put amount eurfrgbp9.2;
5  datalines;
  E71
    E71.42
1,763
E1,763.32
17632
17,632.39

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td></td>
<td>E71</td>
</tr>
<tr>
<td>1234.56</td>
<td></td>
<td>E71.42</td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td>1,763</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E1,763.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17632</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17,632.39</td>
</tr>
</tbody>
</table>

### EURFRGRDw.d Format

Converts an amount from Greek drachmas to euros.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

**EURFRGRDw.d**

#### Syntax Description

- **w**  
  specifies the width of the output field.  
  Default 6

- **d**  
  specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURFRGRD<code>w.d</code> format converts an amount from Greek drachmas to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRGRD<code>w.d</code> format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

#### Example

The following table shows input values in Greek drachmas, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrgrd5.;  
  put amount eurfrgrd9.2;  
datalines;
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrgp5.;</td>
<td>E71</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgp9.2;</td>
<td>E71.42</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrgp5.;</td>
<td>1,763</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgp9.2;</td>
<td>E1,763.32</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrgp5.;</td>
<td>17632</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgp9.2;</td>
<td>17,632.39</td>
</tr>
</tbody>
</table>
EURFRHUFw.d Format

Converts an amount from Hungarian forints to euros.

**Category:**  Currency Conversion  
**Alignment:**  Right

### Syntax

EURFRHUFw.d

#### Syntax Description

**w**  
specifies the width of the output field.  
Default 6  

**d**  
specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURFRHUFw.d format converts an amount from Hungarian forints to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRHUFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in Hungarian forints, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrhuf5.;  
  put amount eurfrhuf9.2;  
  datalines;  
300  
30234.56  
302345  
;  
run;  
SAS log:  
3      put amount eurfrhuf5.;  
4      put amount eurfrhuf9.2;  
5      datalines;  
        E1  
        E1,15  
        E116  
        E116,14  
        1.161  
        E1.161,41  
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>put amount eurfrhuf5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>E1,15</td>
</tr>
<tr>
<td>30234.56</td>
<td>put amount eurfrhuf5.;</td>
<td>E116</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>E116,14</td>
</tr>
<tr>
<td>302345</td>
<td>put amount eurfrhuf5.;</td>
<td>1.161</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>E1.161,41</td>
</tr>
</tbody>
</table>

EURFRIEPw.d Format

Converts an amount from Irish pounds to euros.
Currency Conversion

Syntax

EURFRIEP\textsubscript{w.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default 6

\texttt{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRIEP\textsubscript{w.d} format converts an amount from Irish pounds to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRIEP\textsubscript{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in Irish pounds, SAS statements, and the conversion results in euros.

```sas
data _null_;  
input amount;  
put amount eurfrip5.;  
put amount eurfrip9.2;  
datalines;  
1  
1234.56  
12345  
;  
run;  
8  put amount eurfrip5.;  
9  put amount eurfrip9.2;  
10  datalines;  
E1  
E1.27  
1,568  
E1,567.57  
15675  
15,674.92
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1234.56</td>
<td>E1</td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td>E1.27</td>
</tr>
<tr>
<td>1</td>
<td>1234.56</td>
<td>E1</td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td>E1.27</td>
</tr>
<tr>
<td>1</td>
<td>1234.56</td>
<td>E1</td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td>E1.27</td>
</tr>
<tr>
<td>1</td>
<td>1234.56</td>
<td>E1</td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td>E1.27</td>
</tr>
<tr>
<td>1</td>
<td>1234.56</td>
<td>E1</td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td>E1.27</td>
</tr>
</tbody>
</table>

Appendix 1 • Additional NLS Language Elements
<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurfriep5.;</td>
<td>€1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>€1.27</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfriep5.;</td>
<td>1,568</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>€1,567.57</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfriep5.;</td>
<td>15675</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>15,674.92</td>
</tr>
</tbody>
</table>

**EURFRITLw.d Format**

Converts an amount from Italian lire to euros.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

EURFRITL

**Syntax Description**

- **w** specifies the width of the output field.  
  Default: 6

- **d** specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRITLw.d format converts an amount from Italian lire to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRITLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Italian lire, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrit5.;  
  put amount eurfrit15.;  
  put amount eurfrit19.2;  
datalines;
```
2000
7234.56
72345
;
run;
8 put amount eurfritl5.;
9 put amount eurfritl9.2;
10 datalines;
  E1
  E1,03
  E4
  E3,74
  E37
  E37,36

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>put amount eurfritl5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfritl9.2;</td>
<td>E1,03</td>
</tr>
<tr>
<td>7234.56</td>
<td>put amount eurfritl5.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfritl9.2;</td>
<td>E3,74</td>
</tr>
<tr>
<td>72345</td>
<td>put amount eurfritl5.;</td>
<td>E37</td>
</tr>
<tr>
<td></td>
<td>put amount eurfritl9.2;</td>
<td>E37,36</td>
</tr>
</tbody>
</table>

**EURFRLUFw.d Format**

Converts an amount from Luxembourg francs to euros.

- **Category:** Currency Conversion
- **Alignment:** Right

**Syntax**

```
EURFRLUFw.d
```

**Syntax Description**

- `w`
  - specifies the width of the output field.
  - **Default:** 6

- `d`
  - specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURFRLUFw.d format converts an amount from Luxembourg francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRLUFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in Luxembourg francs, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrluf5.;  
  put amount eurfrluf9.2;  
  datalines;  
50  
1234.56  
12345  
;  
run;  
8  put amount eurfrluf5.;  
9  put amount eurfrluf9.2;  
10  datalines;  
  E1  
  E1,24  
  E31  
  E30,60  
  E306  
  E306,02

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrluf5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E1,24</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrluf5.;</td>
<td>E31</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E30,60</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrluf5.;</td>
<td>E306</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E306,02</td>
</tr>
</tbody>
</table>

EURFRNLGw.d Format

Converts an amount from Dutch guilders to euros.

Category: Currency Conversion
Syntax
EURFRNLG\(w.d\)

Syntax Description
\(w\)

specifies the width of the output field.
Default \(6\)

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

Details
The EURFRNLG\(w.d\) format converts an amount from Dutch guilders to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRNLG\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example
The following table shows input values in Dutch guilders, SAS statements, and the conversion results in euros.

data _null_
   input amount;
   put amount eurfrnlg5.;
   put amount eurfrnlg9.2;
   datalines;
50
1234.56
12345
;
run;

8    put amount eurfrnlg5.;
9    put amount eurfrnlg9.2;
10   datalines;
   E23
   E22,69
   ES60
   ES60,22
   S.602
   ES.601,92

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>--------</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1234.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E22,69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ES60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ES60,22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.602</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ES.601,92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amounts</td>
<td>Statements</td>
<td>Results</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>50</td>
<td>put amount eurfrnlg5.; put amount eurfrnlg9.2;</td>
<td>E23 E22,69</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrnlg5.; put amount eurfrnlg9.2;</td>
<td>E560 E560,22</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrnlg5.; put amount eurfrnlg9.2;</td>
<td>5.602 E5.601,92</td>
</tr>
</tbody>
</table>

**EURFRNOKw.d Format**

Converts an amount from Norwegian krone to euros.

*Category:* Currency Conversion  
*Alignment:* Right

**Syntax**

`EURFRNOKw.d`

**Syntax Description**

`w`

specifies the width of the output field.

Default 6

`d`

specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRNOKw.d format converts an amount from Norwegian krone to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRNOKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Norwegian krone, SAS statements, and the conversion results in euros.

data _null_;  
  input amount;  
  put amount eurfrnok5.;  
  put amount eurfrnok9.2;  
  datalines;
50
1234.56
12345
;
run;
SAS log:
3      put amount eurfrnok5.;
4      put amount eurfrnok9.2;
5      datalines;
      E5
      E5,44
E134
  E134,22
1.342
E1.342,18

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrnok5.;</td>
<td>E5</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnok9.2;</td>
<td>E5,44</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrnok5.;</td>
<td>E134</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnok9.2;</td>
<td>E134,22</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrnok5.;</td>
<td>1.342</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnok9.2;</td>
<td>E1.342,18</td>
</tr>
</tbody>
</table>

**EURFRPLZw.d Format**

Converts an amount from Polish zloty to euros.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

EURFRPLZw.d

**Syntax Description**

w

specifies the width of the output field.

Default 6

d

specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURFRPLZw.d format converts an amount from Polish zloty to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRPLZw.d format and the EUROCURRE function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in Polish zloty, SAS statements, and the conversion results in euros.

```sas
data _null_; input amount; put amount eurfrplz5.; put amount eurfrplz9.2; datalines; 50 1234.56 12345 ; run;
```

SAS log:

```
3   put amount eurfrplz5.;
4   put amount eurfrplz9.2;
5   datalines;
    E12
    E11,90
    E294
    E293,94
    2.939
    E2.939,29
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrplz5.; put amount eurfrplz9.2;</td>
<td>E12 E11,90</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrplz5.; put amount eurfrplz9.2;</td>
<td>E294 E293,94</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrplz5.; put amount eurfrplz9.2;</td>
<td>2.939 E2.939,29</td>
</tr>
</tbody>
</table>

EURFRPTEw.d Format

Converts an amount from Portuguese escudos to euros.
Syntax

EURFRPTE\(w.d\)

**Syntax Description**

\(w\)

specifies the width of the output field.

Default 6

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRPTE\(w.d\) format converts an amount from Portuguese escudos to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRPTE\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Portuguese escudos, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrpte5.;  
  put amount eurfrpte9.2;  
  datalines;  
300  
30234.56  
302345  
;  
run;  
8     put amount eurfrpte5.;  
9     put amount eurfrpte9.2;  
10    datalines;  
   E1  
   E1,50  
E151  
E150,81  
1.508  
E1.508,09
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>8</td>
<td>E1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>E1,50</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>E151</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E150,81</td>
<td></td>
</tr>
<tr>
<td>1.508</td>
<td>E1.508,09</td>
<td></td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>put amount eurfrpte5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E1,50</td>
</tr>
<tr>
<td>30234.56</td>
<td>put amount eurfrpte5.;</td>
<td>E151</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E150,81</td>
</tr>
<tr>
<td>302345</td>
<td>put amount eurfrpte5.;</td>
<td>1.508</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E1.508,09</td>
</tr>
</tbody>
</table>

**EURFRROLw.d Format**

Converts an amount from Romanian lei to euros.

Category: Currency Conversion  
Alignment: Right

**Syntax**

```
EURFRROLw.d
```

**Syntax Description**

- `w` specifies the width of the output field.  
  Default 6

- `d` specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRROLw.d format converts an amount from Romanian lei to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRROLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Romanian lei, SAS statements, and the conversion results in euros.

```sas
data _null_;  
   input amount;  
   put amount eurfrrol5.;  
   put amount eurfrrol9.2;  
   datalines;
```
EURFRRURw.d Format

Converts an amount from Russian rubles to euros.

**Category:** Currency Conversion

**Alignment:** Right

### Syntax

**EURFRRURw.d**

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - Default: 6

- **d**
  - Specifies the number of digits to the right of the decimal point in the numeric value.

### Details

The EURFRRURw.d format converts an amount from Russian rubles to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that...
is incorporated into the EURFRRUR{w,d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in Russian rubles, SAS statements, and the conversion results in euros.

```sas
data _null_;  
  input amount;  
  put amount eurfrrur5.;  
  put amount eurfrrur9.2;  
  datalines;  
50  
5234.56  
52345  
;  
run;  
  E3  
  E2,53  
E265  
E264,80  
2.648  
E2.647,97
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrrur5.;</td>
<td>E3</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E2,53</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrrur5.;</td>
<td>E265</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E264,80</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrrur5.;</td>
<td>2.648</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E2.647,97</td>
</tr>
</tbody>
</table>

**EURFRSEK{w,d} Format**

Converts an amount from Swedish kronor to euros.

- **Category:** Currency Conversion
- **Alignment:** Right

**Syntax**

EURFRSEK{w,d}
Syntax Description

\( w \)

specifies the width of the output field.

Default 6

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRSEK\( w,d \) format converts an amount from Swedish kronor to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRSEK\( w,d \) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in Swedish kronor, SAS statements, and the conversion results in euros.

data _null_
  input amount;
  put amount eurfrsek5.;
  put amount eurfrsek9.2;
  datalines;
  50
  1234.56
  12345
  ;
run;

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrsek5.;</td>
<td>E5</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E5,34</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrsek5.;</td>
<td>E132</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E131,81</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrsek5.;</td>
<td>1.318</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E1.318,08</td>
</tr>
</tbody>
</table>
EURFRSITw.d Format

Converts an amount from Slovenian tolars to euros.

Category: Currency Conversion
Alignment: Right

Syntax

EURFRSITw.d

Syntax Description

w

specifies the width of the output field.

Default 6

d

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRSITw.d format converts an amount from Slovenian tolars to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRSITw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Note: Slovenia's currency is the Euro. The information for EURFRSIT is provided for user's historical data.

Example

The following table shows input values in Slovenian tolars, SAS statements, and the conversion results in euros.

data _null_
  input amount;
  put amount eurfrsit5.;
  put amount eurfrsit9.2;
  datalines;
200
20234.56
202345
;
run;

E1
E1,05
E106
E105,94
1.059
E1.059,40
EURFRTRLw.d Format

Converts an amount from Turkish liras to euros.

**Category:** Currency Conversion  
**Alignment:** Right

### Syntax

```
EURFRTRLw.d
```

### Syntax Description

- `w` specifies the width of the output field.  
  Default: 6

- `d` specifies the number of digits to the right of the decimal point in the numeric value.

### Details

The EURFRTRLw.d format converts an amount from Turkish liras to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRTRLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

### Example

The following table shows input values in Turkish liras, SAS statements, and the conversion results in euros.

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>put amount eurfrsit5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsit9.2;</td>
<td>E1,05</td>
</tr>
<tr>
<td>20234.56</td>
<td>put amount eurfrsit5.;</td>
<td>E106</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsit9.2;</td>
<td>E105,94</td>
</tr>
<tr>
<td>202345</td>
<td>put amount eurfrsit5.;</td>
<td>1.059</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsit9.2;</td>
<td>E1.059,40</td>
</tr>
</tbody>
</table>

```sas
data _null_;  
  input amount;  
  put amount eurfrtrl5.;  
```

```
```
put amount eurfrtrl9.2;
datalines;
400
40234.56
402345
;
run;

   E1
   E1, 19
   E119
   E119, 42
E1. 194
E1. 194, 21

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>put amount eurfrtrl5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtrl9.2;</td>
<td>E1, 19</td>
</tr>
<tr>
<td>40234.56</td>
<td>put amount eurfrtrl5.;</td>
<td>E119</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtrl9.2;</td>
<td>E119, 42</td>
</tr>
<tr>
<td>402345</td>
<td>put amount eurfrtrl5.;</td>
<td>1.194</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtrl9.2;</td>
<td>E1. 194, 21</td>
</tr>
</tbody>
</table>

**EURTOATS**<sub>w.d</sub> Format

Converts an amount from euros to Austrian schillings.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

`EURTOATS w.d`

**Syntax Description**

`w`

specifies the width of the output field.

*Default* 6

`d`

specifies the number of digits to the right of the decimal point in the numeric value.
Details
The EURTOATSw.d format converts an amount in euros to an amount in Austrian schillings. The conversion rate is a fixed rate that is incorporated into the EURTOATSw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example
The following table shows input values in euros, SAS statements, and the conversion results in Austrian schillings.

```sas
data _null_;  
   input amount;  
   put amount eurtoats6.;  
   put amount eurtoats12.2;  
   datalines;  
1 1234.56  
12345 ;  
run;  
80 put amount eurtoats6.;  
81 put amount eurtoats12.2;  
82 datalines;  
14 13.76  
16988  
16987.92  
169871  
169870.90
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoats6.;</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoats12.2;</td>
<td>13.76</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoats6.;</td>
<td>16988</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoats12.2;</td>
<td>16987.92</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoats6.;</td>
<td>169871</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoats12.2;</td>
<td>169870.90</td>
</tr>
</tbody>
</table>

EURTOBEFw.d Format
Converts an amount from euros to Belgian francs.

Category: Currency Conversion
Syntax

EURTOBEF\textit{w}\textit{.d}

\textbf{Syntax Description}

\textit{w}

specifies the width of the output field.

Default 6

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

\textbf{Details}

The EURTOBEF\textit{w}\textit{.d} format converts an amount in euros to an amount in Belgian francs. The conversion rate is a fixed rate that is incorporated into the EURTOBEF\textit{w}\textit{.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

\textbf{Example}

The following table shows input values in euros, SAS statements, and the conversion results in Belgian francs.

```sas
data _null_
    input amount;
    put amount eurtobef6.;
    put amount eurtobef12.2;
    datalines;
1 1234.56
1234 56 
;
run;
8       put amount eurtobef6.;
9       put amount eurtobef12.2;
10      datalines;
40
40.34
49802
49802.03
497996
497996.07
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1234.56</td>
<td></td>
</tr>
<tr>
<td>1234</td>
<td>12345</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>40.34</td>
<td></td>
</tr>
<tr>
<td>49802</td>
<td>49802.03</td>
<td></td>
</tr>
<tr>
<td>497996</td>
<td>497996.07</td>
<td></td>
</tr>
</tbody>
</table>

---

1

2
EURTOCHFw.d Format

Converts an amount from euros to Swiss francs.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

EURTOCHFw.d

**Syntax Description**

- \( w \)
  - specifies the width of the output field.
  - Default 6

- \( d \)
  - specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOCHFw.d format converts an amount in euros to an amount in Swiss francs. The conversion rate is a changeable rate that is incorporated into the EURTOCHFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in euros, SAS statements, and the conversion results in Swiss francs.

```sas
data _null_
  input amount;
  put amount eurtochf6.;
  put amount eurtochf12.2;
datalines;
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtochf6.;</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>put amount eurtochf12.2;</td>
<td>40.34</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtochf6.;</td>
<td>49802</td>
</tr>
<tr>
<td></td>
<td>put amount eurtochf12.2;</td>
<td>49802.03</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtochf6.;</td>
<td>497996</td>
</tr>
<tr>
<td></td>
<td>put amount eurtochf12.2;</td>
<td>497996.07</td>
</tr>
</tbody>
</table>
```
EURTOCZKw.d Format

Converts an amount from euros to Czech koruny.

Category: Currency Conversion
Alignment: Right

Syntax

EURTOCZKw.d

Syntax Description

w
  specifies the width of the output field.
  Default  6

d
  specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURTOCZKw.d format converts an amount in euros to an amount in Czech koruny. The conversion rate is a changeable rate that is incorporated into the EURTOCZKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Czech koruny.

data _null_;  
  input amount;  
  put amount eurtoczk6.;  
  put amount eurtoczk12.2;  
  datalines;  
  1  
  1234.56  
  12345  
  ;  
  run;  
SAS log:  
104     put amount eurtoczk6.;  
105     put amount eurtoczk12.2;  
106     datalines;  
    35  
    34.86  
43032  
43032.19  
430301  
430301.02

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoczk6.;</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoczk12.2;</td>
<td>34.86</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoczk6.;</td>
<td>43032</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoczk12.2;</td>
<td>43032.19</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoczk6.;</td>
<td>430301</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoczk12.2;</td>
<td>430301.02</td>
</tr>
</tbody>
</table>

EURTODEMW.d Format

Converts an amount from euros to Deutsche marks.
Category: Currency Conversion
Alignment: Right

Syntax

EURTODEM<sub>w.d</sub>

Syntax Description

- <sub>w</sub>
  - specifies the width of the output field.
  - Default: 6

- <sub>d</sub>
  - specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTODEM<sub>w.d</sub> format converts an amount in euros to an amount in Deutsche marks. The conversion rate is a fixed rate that is incorporated into the EURTODEM<sub>w.d</sub> format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Deutsche marks.

```
data _null_;  
  input amount;  
  put amount eurtodem6.;  
  put amount eurtodem12.2;  
  datalines;  
1 1234.56  
12345  
;  
run;  
8 put amount eurtodem6.;  
9 put amount eurtodem12.2;  
10 datalines;  
2 1.96  
2415  
2414.59  
24145  
24144.72```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234.56</td>
<td>put amount eurtodem6.;</td>
<td>1431.64</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtodem12.2;</td>
<td>14315.95</td>
</tr>
</tbody>
</table>

…”Currency Representation” on page 69.
<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtodem6.; put amount eurtodem12.2;</td>
<td>2 1.96</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtodem6.; put amount eurtodem12.2;</td>
<td>2415 2414.59</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtodem6.; put amount eurtodem12.2;</td>
<td>24145 24144.72</td>
</tr>
</tbody>
</table>

**EURTODKKw.d Format**

Converts an amount from euros to Danish kroner.

- **Category:** Currency Conversion
- **Alignment:** Right

**Syntax**

\[
\text{EURTODKK}w.d
\]

**Syntax Description**

- **w** specifies the width of the output field.
  - Default 6
- **d** specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTODKKw.d format converts an amount in euros to an amount in Danish kroner. The conversion rate is a changeable rate that is incorporated into the EURTODKKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in euros, SAS statements, and the conversion results in Danish kroner.

```sas
data _null_;  
  input amount;  
  put amount eurtodkk6.;  
  put amount eurtodkk12.2;  
  datalines;
```
1
1234.56
12345
;
run;
SAS log:
62 put amount eurtodkk6.;
63 put amount eurtodkk12.2;
64 datalines;

7
7.49
9247
9246.97
92465
92465.16

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtodkk6.;</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>7.49</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtodkk6.;</td>
<td>9247</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>9246.97</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtodkk6.;</td>
<td>92465</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>92465.16</td>
</tr>
</tbody>
</table>

**EURTOESPw.d Format**

Converts an amount from euros to Spanish peseta.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

```plaintext
EURTOESPw.d
```

**Syntax Description**

- `w`  
  specifies the width of the output field.  
  Default 6

- `d`  
  specifies the number of digits to the right of the decimal point in the numeric value.
Details
The EURTOESPw.d format converts an amount in euros to an amount in Spanish peseta. The conversion rate is a fixed rate that is incorporated into the EURTOESPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example
The following table shows input values in euros, SAS statements, and the conversion results in Spanish peseta.

data _null_
  input amount;
  put amount eurtoesp8.;
  put amount eurtoesp12.2;
  datalines;
  1 1234.56
  12345
; run;
26 put amount eurtoesp8.;
27 put amount eurtoesp12.2;
28 datalines;
  166
  166.39
  205414
  205413.50
  2054035
  2054035.17

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoesp8.;</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoesp12.2;</td>
<td>166.39</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoesp8.;</td>
<td>205414</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoesp12.2;</td>
<td>205413.50</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoesp8.;</td>
<td>2054035</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoesp12.2;</td>
<td>2054035.17</td>
</tr>
</tbody>
</table>

EURTOFIMw.d Format
Converts an amount from euros to Finnish markkas.

Category: Currency Conversion
Syntax

EURTOFIMw,d

Syntax Description

w
specifies the width of the output field.
Default  6

d
specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOFIMw,d format converts an amount in euros to an amount in Finnish markkas. The conversion rate is a fixed rate that is incorporated into the EURTOFIMw,d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Finnish markkas.

data _null_;  
  input amount;  
  put amount eurtofim6.;  
  put amount eurtofim12.2;  
 datalines;  
1  
1234.56  
12345  
;  
run;  
8      put amount eurtofim6.;  
9      put amount eurtofim12.2;  
10     datalines;  
6  
5.95  
7340  
7340.36  
73400  
73400.04  

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1234.56</td>
<td>12345</td>
</tr>
<tr>
<td>1234.56</td>
<td>1234</td>
<td>12345</td>
</tr>
<tr>
<td>12345</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.95</td>
<td>7340</td>
<td>7340</td>
</tr>
<tr>
<td>7340</td>
<td>7340.36</td>
<td>7340</td>
</tr>
<tr>
<td>73400</td>
<td>73400.04</td>
<td>73400</td>
</tr>
</tbody>
</table>
### EURTOFRFw.d Format

Converts an amount from euros to French francs.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

EURTOFRFw.d

#### Syntax Description

- **w**  
  Specifies the width of the output field.  
  Default 6

- **d**  
  Specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTOFRFw.d format converts an amount in euros to an amount in French francs. The conversion rate is a fixed rate that is incorporated into the EURTOFRFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

#### Example

The following table shows input values in euros, SAS statements, and the conversion results in French francs.

```sas
data _null_;  
  input amount;  
  put amount eurtofrf6.;  
  put amount eurtofrf12.2;  
datalines;
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtofrf6.;</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>5.95</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtofrf6.;</td>
<td>7340</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>7340.36</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtofrf6.;</td>
<td>73400</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>73400.04</td>
</tr>
</tbody>
</table>
1
1234.56
12345
;
run;
8       put amount eurtofrf6.;
9       put amount eurtofrf12.2;
10      datalines;
    7
6.56
8098
8098.18
80978
80977.89

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtofrf6.;</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>6.56</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtofrf6.;</td>
<td>8098</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>8098.18</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtofrf6.;</td>
<td>80978</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>80977.89</td>
</tr>
</tbody>
</table>

**EURTOGBPw.d Format**

Converts an amount from euros to British pounds.

- **Category:** Currency Conversion
- **Alignment:** Right

**Syntax**

```
EURTOGBPw.d
```

**Syntax Description**

- `w`
  - specifies the width of the output field.
  - Default: 6

- `d`
  - specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURTOGBP<sub>w.d</sub> format converts an amount in euros to an amount in British pounds. The conversion rate is a changeable rate that is incorporated into the EURTOGBP<sub>w.d</sub> format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in British pounds.

```sas
data _null_;  
   input amount;  
   put amount eurtogbp6.;  
   put amount eurtogbp12.2;  
   datalines;  
1  
1234.56  
12345  
;  
runit;  
SAS log:  
8       put amount eurtogbp6.;  
9       put amount eurtogbp12.2;  
10      datalines;  
1       0.70  
864  
864.35  
8643  
8643.13
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtogbp6.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>0.70</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtogbp6.;</td>
<td>864</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>864.35</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtogbp6.;</td>
<td>8643</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>8643.13</td>
</tr>
</tbody>
</table>

EURTOGRD<sub>w.d</sub> Format

Converts an amount from euros to Greek drachmas.
**Syntax**

EURTOGRD<sup>w</sup><sub>d</sub>

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - Default: 6

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOGRD<sup>w</sup><sub>d</sub> format converts an amount in euros to an amount in Greek drachmas. The conversion rate is a fixed rate that is incorporated into the EURTOGRD<sup>w</sup><sub>d</sub> format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in euros, SAS statements, and the conversion results in Greek drachmas.

```sas
data _null_;   input amount;   put amount eurtogr8.;   put amount eurtogr16.2;   datalines;  1  1234.56  12345  ; run; SAS log: 65 put amount eurtogr8.; 66 put amount eurtogr16.2; 67 datalines;  341  340.89  420843  420842.99  4208225  4208225.33
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
### EURTOHUF\(w,d\) Format

Converts an amount from euros to Hungarian forints.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

**EURTOHUF\(w,d\)**

#### Syntax Description

- **\(w\)** specifies the width of the output field.  
  - **Default:** 6
- **\(d\)** specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTOHUF\(w,d\) format converts an amount in euros to an amount in Hungarian forints. The conversion rate is a changeable rate that is incorporated into the EURTOHUF\(w,d\) format and the EUROCURRE function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

#### Example

The following table shows input values in euros, SAS statements, and the conversion results in Hungarian forints.

```sas
data _null_;  
  input amount;  
  put amount eurtohuf8.;  
  put amount eurtohuf14.2;  
  datalines;
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtohuf8.;</td>
<td>341</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf16.2;</td>
<td>340.89</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtohuf8.;</td>
<td>420843</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf16.2;</td>
<td>420842.99</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtohuf8.;</td>
<td>4208225</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf16.2;</td>
<td>4208225.33</td>
</tr>
</tbody>
</table>
1
1234.56
12345
;
run;
SAS log:
140     put amount eurtohuf8.;
141     put amount eurtohuf14.2;
142     datalines;
       260
       260.33
       321387
       321386.83
       3213712
       3213712.13

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtohuf8.;</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf14.2;</td>
<td>260.33</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtohuf8.;</td>
<td>321387</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf14.2;</td>
<td>321386.83</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtohuf8.;</td>
<td>3213712</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf14.2;</td>
<td>3213712.13</td>
</tr>
</tbody>
</table>

**EURTOIEPw.d Format**

Converts an amount from euros to Irish pounds.

**Category:** Currency Conversion

**Alignment:** Right

**Syntax**

EURTOIEP\(w.d\)

**Syntax Description**

\(w\)

specifies the width of the output field.

Default \(6\)

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.
Details

The EURTOIEPw.d format converts an amount in euros to an amount in Irish pounds. The conversion rate is a fixed rate that is incorporated into the EURTOIEPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Irish pounds.

data _null_;  
  input amount;  
  put amount eurtoiep6.;  
  put amount eurtoiep12.2;  
  datalines;  
1 1234.56  
12345  
;  
run;  
8 put amount eurtoiep6.;  
9 put amount eurtoiep12.2;  
10 datalines;  
1 0.79  
972  
972.30  
9722  
9722.48

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoiep6.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>0.79</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoiep6.;</td>
<td>972</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>972.30</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoiep6.;</td>
<td>9722</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>9722.48</td>
</tr>
</tbody>
</table>

EURTOITLw.d Format

Converts an amount from euros to Italian lire.

Category: Currency Conversion
Syntax
EURTOITL\(w.d\)

**Syntax Description**

\(w\)

specifies the width of the output field.

Default 6

\(d\)

specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOITL\(w.d\) format converts an amount in euros to an amount in Italian lire. The conversion rate is a fixed rate that is incorporated into the EURTOITL\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in euros, SAS statements, and the conversion results in Italian lire.

data _null_;  
input amount;  
put amount eurtoitl8.;  
put amount eurtoitl12.2;  
datalines;  
1  
1234.56  
12345  
;  
rung;  
44 put amount eurtoitl8.;  
45 put amount eurtoitl12.2;  
46 datalines;  
1936  
1936.27  
2390441  
2390441.49  
23903253  
23903253.15

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-----------</td>
<td>1936</td>
</tr>
<tr>
<td>1</td>
<td>put amount eurtoitl8.;</td>
<td>1936</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoitl12.2;</td>
<td>1936.27</td>
</tr>
</tbody>
</table>
### EURTOLUFw.d Format

Converts an amount from euros to Luxembourg francs.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

```
EURTOLUFw.d
```

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - **Default:** 6

- **d**
  - specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOLUFw.d format converts an amount in euros to an amount in Luxembourg francs. The conversion rate is a fixed rate that is incorporated into the EURTOLUFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

**Example**

The following table shows input values in euros, SAS statements, and the conversion results in Luxembourg francs.

```sas
data _null_;  
  input amount;  
  put amount eurtoluf8.;  
  put amount eurtoluf12.2;  
  datalines;  
1  
1234.56  
12345  
;  
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234.56</td>
<td>put amount eurtoluf8.;</td>
<td>2390441</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>2390441.49</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoluf8.;</td>
<td>23903253</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>23903253.15</td>
</tr>
</tbody>
</table>
```
run;
8   put amount eurtoluf6.;
9   put amount eurtoluf12.2;
10  datalines;
40
   40.34
49802
   49802.03
497996
   497996.07

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoluf6.;</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>40.34</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoluf6.;</td>
<td>49802</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>49802.03</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoluf6.;</td>
<td>497996</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>497996.07</td>
</tr>
</tbody>
</table>
```

### EURONLGlw.d Format

Converts an amount from euros to Dutch guilders.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

```
EURONLGlw.d
```

#### Syntax Description

- **w**  
  specifies the width of the output field.  
  
  Default: 6

- **d**  
  specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURONLGlw.d format converts an amount in euros to an amount in Dutch guilders. The conversion rate is a fixed rate that is incorporated into the EURONLGlw.d format and the EUROCURR function. For more information about European currency.
conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Dutch guilders.

```sas
data _null_;  
  input amount;  
  put amount eurtonlg6.;  
  put amount eurtonlg12.2;  
  datalines;  
  1  
  1234.56  
  12345  
  ;  
  run;  
  8       put amount eurtonlg6.;  
  9       put amount eurtonlg12.2;  
  10      datalines;  
        2  
        2.20  
        2721  
        2720.61  
        27205  
        27204.80
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtonlg6.;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonlg12.2;</td>
<td>2.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtonlg6.;</td>
<td>2721</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonlg12.2;</td>
<td>2720.61</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtonlg6.;</td>
<td>27205</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonlg12.2;</td>
<td>27204.80</td>
</tr>
</tbody>
</table>

**EURTONOKw.d Format**

Converts an amount from euros to Norwegian krone.

- **Category:** Currency Conversion
- **Alignment:** Right
Syntax

EURTONOK\( w.d \)

Syntax Description

\( w \)

specifies the width of the output field.

Default 6

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTONOK\( w.d \) format converts an amount in euros to an amount in Norwegian krone. The conversion rate is a changeable rate that is incorporated into the EURTONOK\( w.d \) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Norwegian krone.

```sas
data _null_;  
  input amount;  
  put amount eurtonok6.;  
  put amount eurtonok12.2;  
  datalines;  
1  
1234.56  
12345  
;  
run;  
SAS log:  
158 put amount eurtonok6.;  
159 put amount eurtonok12.2;  
160 datalines;  
  9  
9.20  
11355  
11355.11  
113546  
113545.61
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtonok6.; put amount eurtonok12.2;</td>
<td>9.20</td>
</tr>
</tbody>
</table>
### EURTOPLZw.d Format

Converts an amount from euros to Polish zloty.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

EURTOPLZ\(w.d\)

#### Syntax Description

\(w\)
- specifies the width of the output field.  
  - Default 6

\(d\)
- specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTOPLZ\(w.d\) format converts an amount in euros to an amount in Polish zloty. The conversion rate is a changeable rate that is incorporated into the EURTOPLZ\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

#### Example

The following table shows input values in euros, SAS statements, and the conversion results in Polish zloty.

```sas
data _null_;  
  input amount;  
  put amount eurtoplz6.;  
  put amount eurtoplz12.2;  
  datalines;  
1  
1234.56  
12345  
12345.61
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234.56</td>
<td>put amount eurtonok6.;</td>
<td>11355</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2;</td>
<td>11355.11</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtonok6.;</td>
<td>113546</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2;</td>
<td>113545.61</td>
</tr>
</tbody>
</table>
run;
SAS log:
80      put amount eurtoplz6.;
81      put amount eurtoplz12.2;
82      datalines;
4        4.20
5185    5185.15
51849   51849.00

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>----+----1----2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>put amount eurtoplz6.;</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplz12.2;</td>
<td>4.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoplz6.;</td>
<td>5185</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplz12.2;</td>
<td>5185.15</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoplz6.;</td>
<td>51849</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplz12.2;</td>
<td>51849.00</td>
</tr>
</tbody>
</table>

**EURTOPE\textsubscript{w.d} Format**

Converts an amount from euros to Portuguese escudos.

**Category:** Currency Conversion  
**Alignment:** Right

**Syntax**

EURTOPE\textsubscript{w.d}

**Syntax Description**

\textit{w}

specifies the width of the output field.

Default 6

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOPE\textsubscript{w.d} format converts an amount in euros to an amount in Portuguese escudos. The conversion rate is a fixed rate that is incorporated into the EURTOPE\textsubscript{w.d}
format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Portuguese escudos.

data _null_;  
   input amount;  
   put amount europte8.;  
   put amount europte12.2;  
   datalines;  
1  
1234.56  
12345  
;  
run;  
26   put amount europte8.;  
27   put amount europte12.2;  
28   datalines;  
200  
200.48  
247507  
247507.06  
2474950  
2474950.29

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount europte8.;</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>put amount europte12.2;</td>
<td>200.48</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount europte8.;</td>
<td>247507</td>
</tr>
<tr>
<td></td>
<td>put amount europte12.2;</td>
<td>247507.06</td>
</tr>
<tr>
<td>12345</td>
<td>put amount europte8.;</td>
<td>2474950</td>
</tr>
<tr>
<td></td>
<td>put amount europte12.2;</td>
<td>2474950.29</td>
</tr>
</tbody>
</table>

EURTOROLOw.d Format

Converts an amount from euros to Romanian lei.

Category: Currency Conversion

Alignment: Right
Syntax

EURTOROL\textit{w}.\textit{d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default 6

\textit{d}

specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOROL\textit{w}.\textit{d} format converts an amount in euros to an amount in Romanian lei. The conversion rate is a changeable rate that is incorporated into the EURTOROL\textit{w}.\textit{d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Romanian lei.

data _null_

   input amount;
   put amount eurtorol6.;
   put amount eurtorol12.2;
   datalines;
   1
   1234.56
   12345
; run;
SAS log:

98   put amount eurtorol6.;
99   put amount eurtorol12.2;
100  datalines;

   14
   13.71
   16926
   16925.82
   169250
   169249.95

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtorol6.;</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorol12.2;</td>
<td>13.71</td>
</tr>
</tbody>
</table>
### EURTORURw.d Format

Converts an amount from euros to Russian rubles.

**Category:** Currency Conversion  
**Alignment:** Right

#### Syntax

\[ \text{EURTORUR} \text{w}.d \]

#### Syntax Description

- \( w \) specifies the width of the output field.  
  Default 6

- \( d \) specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTORURw.d format converts an amount in euros to an amount in Russian rubles. The conversion rate is a changeable rate that is incorporated into the EURTORURw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

#### Example

The following table shows input values in euros, SAS statements, and the conversion results in Russian rubles.

```sas
data _null_;  
  input amount;  
  put amount eurtor6.;  
  put amount eurtor12.2;  
  datalines;  
1 1234.56 12345
12345
;```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234.56</td>
<td>put amount eurtor6.;</td>
<td>16926</td>
</tr>
<tr>
<td></td>
<td>put amount eurtor12.2;</td>
<td>16925.82</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtor6.;</td>
<td>169250</td>
</tr>
<tr>
<td></td>
<td>put amount eurtor12.2;</td>
<td>169249.95</td>
</tr>
</tbody>
</table>
run;
SAS log:
8     put amount eurtorur6.;
9     put amount eurtorur12.2;
10    datalines;
   20
   19.77
24405
   24404.78
244036
   244035.96

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtorur6.;</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>19.77</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtorur6.;</td>
<td>24405</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>24404.78</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtorur6.;</td>
<td>244036</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>244035.96</td>
</tr>
</tbody>
</table>

EURTOSEKw.d Format

Converts an amount from euros to Swedish kronor.

Category: Currency Conversion
Alignment: Right

Syntax

EURTOSEKw.d

Syntax Description

w
specifies the width of the output field.

Default 6

_d
specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOSEKw.d format converts an amount in euros to an amount in Swedish kronor. The conversion rate is a changeable rate that is incorporated into the
EURTOSEKw.d format and the EUROCURRE function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Example

The following table shows input values in euros, SAS statements, and the conversion results in Swedish kronor.

data _null_;  
  input amount;  
  put amount eurtosek6.;  
  put amount eurtosek12.2;  
  datalines;  
  1  
  1234.56  
  12345  
  ;  
run;  
SAS log:  
86      put amount eurtosek6.;  
87      put amount eurtosek12.2;  
88      datalines;  
9      9.37  
       11563  
       11562.78  
       115622  
       115622.16

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>put amount eurtosek6.;</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosek12.2;</td>
<td>9.37</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtosek6.;</td>
<td>11563</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosek12.2;</td>
<td>11562.78</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtosek6.;</td>
<td>115622</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosek12.2;</td>
<td>115622.16</td>
</tr>
</tbody>
</table>

EURTOSITw.d Format

Converts an amount from euros to Slovenian tolers.

Category: Currency Conversion

Alignment: Right
Syntax
EURTOSITw.d

Syntax Description
w
  specifies the width of the output field.
  Default  6

d
  specifies the number of digits to the right of the decimal point in the numeric value.

Details
The EURTOSITw.d format converts an amount in euros to an amount in Slovenian tolars. The conversion rate is a changeable rate that is incorporated into the EURTOSITw.d format and the EUROCURRE function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

Note: Slovenia's currency is the Euro. The information for EURTOSIT is provided for user's historical data.

Example
The following table shows input values in euros, SAS statements, and the conversion results in Slovenian tolars.

data _null_;  
  input amount;  
  put amount eurtosit8.;  
  put amount eurtosit14.2;  
  datalines;  
1 1234.56 12345  
191 191.00 235801 235800.96 2357895 2357895.00

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---+---+---

### EURTOTRL\(w.d\) Format

Converts an amount from euros to Turkish liras.

**Category:** Currency Conversion

**Alignment:** Right

#### Syntax

EURTOTRL\(w.d\)

#### Syntax Description

- \(w\)
  - Specifies the width of the output field.
  - **Default:** 6

- \(d\)
  - Specifies the number of digits to the right of the decimal point in the numeric value.

#### Details

The EURTOTRL\(w.d\) format converts an amount in euros to an amount in Turkish liras. The conversion rate is a changeable rate that is incorporated into the EURTOTRL\(w.d\) format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Currency Representation” on page 69.

#### Example

The following table shows input values in euros, SAS statements, and the conversion results in Turkish liras.

```sas
data _null_;  
  input amount;  
  put amount eurtotrl8.;  
  put amount eurtotrl14.2;  
datalines;
```

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtotrl8.;</td>
<td>191</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>191.00</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtotrl8.;</td>
<td>235801</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>235800.96</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtotrl8.;</td>
<td>2357895</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>2357895.00</td>
</tr>
</tbody>
</table>
1
1234.56
12345
;
run;
SAS log:
62 put amount eurtotrl8.;
63 put amount eurtotrl14.2;
64 datalines;
  337
  336.91
415938
  415938.08
4159179
  4159178.64

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtotrl8.;</td>
<td>337</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>336.91</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtotrl8.;</td>
<td>415938</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>415938.08</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtotrl8.;</td>
<td>4159179</td>
</tr>
<tr>
<td></td>
<td>put amount eurtotrl14.2;</td>
<td>4159178.64</td>
</tr>
</tbody>
</table>

**EURDFDEw. Informat**

Reads international date values.

*Category:* Date and Time

**Syntax**

**EURDFDEw:**

**Required Argument**

*w*

specifies the width of the input field.

*Default* 7 (except Finnish)

*Range* 7–32 (except Finnish)

*Note* If you use the Finnish (FIN) language prefix, the w range is 10–32 and the default w is 10.
Details

The date values must be in the form \textit{ddmmmyy} or \textit{ddmmmyyyy}:

\begin{itemize}
\item \textit{dd} is an integer from 01–31 that represents the day of the month.
\item \textit{mmm} is the first three letters of the month name.
\item \textit{yy} or \textit{yyyy} is a two-digit or four-digit integer that represents the year.
\end{itemize}

You can place blanks and other special characters between day, month, and year values.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the DFLANG= system option.

Example

This INPUT statement uses the value of the DFLANG= system option to read the international date values in Spanish.

\begin{verbatim}
options dflang=spanish;
input day eurdfde10.;
\end{verbatim}

This INPUT statement uses the Spanish language prefix in the informat to read the international date values in Spanish. The value of the DFLANG= option, therefore, is ignored.

\begin{verbatim}
input day espdfde10.;
options dflang=spanish;
\end{verbatim}

\begin{verbatim}
data _null_;
input day eurdfde10.;
put day;
datalines;
01abr1999
01-abr-99
;
\end{verbatim}

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>01abr1999</td>
<td>14335</td>
</tr>
<tr>
<td>01-abr-99</td>
<td>14335</td>
</tr>
</tbody>
</table>

\textbf{EURDFDTw. Informat}

Reads international datetime values in the form \textit{ddmmmyy \textit{hh:mm:ss.ss}} or \textit{ddmmmyyyy \textit{hh:mm:ss.ss}}.
Syntax

**EURDFDT**

**Syntax Description**

\( w \)

- specifies the width of the input field.

  Default 18

  Range 13–40

**Details**

The date values must be in the form `ddmmmyy` or `ddmmmyyyy`, followed by a blank or special character, and then the time values as `hh:mm:ss.ss`. The syntax for the date is represented as follows:

- **dd** is an integer from 01–31 that represents the day of the month.
- **mmm** is the first three letters of the month name.
- **yy** or **yyyy** is a two-digit or four-digit integer that represents the year.

The syntax for time is represented as follows:

- **hh** is the number of hours ranging from 00–23,
- **mm** is the number of minutes ranging from 00–59,
- **ss.ss** is the number of seconds ranging from 00–59 with the fraction of a second following the decimal point.

The **EURDFDT**. informat requires values for both the date and the time. However, the **ss.ss** portion is optional.

**Note:** SAS interprets a two-digit year as belonging to the 100-year span that is defined by the **YEARCUTOFF=** system option.

You can set the language for the SAS session with the **DFLANG=** system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “**DFLANG= System Option: UNIX, Windows, and z/OS**” on page 585 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the **DFLANG=** system option.

**Example**

This **INPUT** statement uses the value of the **DFLANG=** system option to read the international datetime values in German.
options dflang=german;
input date eurdfdt20.;

This INPUT statement uses the German language prefix to read the international datetime values in German. The value of the DFLANG= option, therefore, is ignored.

input date deudfdt20.;
options dflang=german;
data _null_;
input date eurdfdt20.;
put date;
datalines;
23dez99:10:03:17.2
23dez1999:10:03:17.2
;

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>23dez99:10:03:17.2</td>
<td>1261562597.2</td>
</tr>
<tr>
<td>23dez1999:10:03:17.2</td>
<td>1261562597.2</td>
</tr>
</tbody>
</table>

**EURDFMYw. Informat**

Reads month and year date values in the form *mmmyy* or *mmmyyyy*.

**Category:** Date and Time

**Syntax**

EURDFMYw.

**Syntax Description**

*w*

specifies the width of the input field.

**Default**

5 (except Finnish)

**Range**

5–32 (except Finnish)

**Note**

If you use the Finnish (FIN) language prefix, the *w* range is 7–32 and the default value for *w* is 7.

**Details**

The date values must be in the form *mmmyy* or *mmmyyyy*:

*mmm*

is the first three letters of the month name.
**yy or yyyy**

is a two-digit or four-digit integer that represents the year.

You can place blanks and other special characters between day, month, and year values. A value that is read with EURDFMYw. results in a SAS date value that corresponds to the first day of the specified month.

*Note:* SAS interprets a two-digit year as belonging to the 100-year span that is defined by the YEARCUTOFF= system option.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: UNIX, Windows, and z/OS” on page 585 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the DFLANG= option.

**Example**

This INPUT statement uses the value of DFLANG= system option to read the international date values in French.

```sas
options dflang=french;
input month eurdfmy7.;
```

The second INPUT statement uses the French language prefix, and DFLANG is not specified.

```sas
input month fradfmy7.;
options dflang=french;
data _null_;  
  input month eurdfmy7.;  
  put month;  
datalines;
avr1999  
  avr 99
;  
options dflang=english;
data _null_;  
  input month fradfmy7.;  
  put month;  
datalines;
avr1999  
  avr 99
;  
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>avr1999</td>
<td>14335</td>
</tr>
<tr>
<td>avr 99</td>
<td>14335</td>
</tr>
</tbody>
</table>
EUROCURR Function

Converts one European currency to another.

Category: Currency Conversion

Syntax

EUROCURR(from-currency-amount, from-currency-code, to-currency-code)

Required Arguments

from-currency-amount
is a numeric value that specifies the amount to convert.

from-currency-code
specifies a three-character currency code that identifies the currency that you are converting from. (See European Currency and Currency Codes on page 816.)

Tip If from-currency-code has a blank value, EUROCURR converts currency values from euros to the currency of the European country that you specify.

See “Example 4: Converting Currency When One Variable Is Blank” on page 818

to-currency-code
specifies a three-character currency code that identifies the currency that you are converting to. (See European Currency and Currency Codes on page 816.)

Tip If to-currency-code has a blank value, EUROCURR converts values from the currency of the European country that you specify to euros.

Details

The following table lists European currencies and the associated currency codes. Use the currency codes to identify the type of currency that you are converting to or converting from. Several countries use the Euro as their currency instead of the currency listed in the following table. This information is provided in order to satisfy user's historical data.

<table>
<thead>
<tr>
<th>Currency</th>
<th>Currency code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austrian schilling</td>
<td>ATS</td>
</tr>
<tr>
<td>Belgian franc</td>
<td>BEF</td>
</tr>
<tr>
<td>British pound sterling</td>
<td>GBP</td>
</tr>
<tr>
<td>Czech koruna</td>
<td>CZK</td>
</tr>
</tbody>
</table>
The EUROCURR function converts a specific country's currency to an equivalent amount in another country's currency. It can also convert a specific country's currency to euros. EUROCURR uses the values in either the fixed currency conversion rate table or the changeable currency conversion rate table to convert currency.

If you are converting from one country's currency to euros, SAS divides by that\textit{from-currency-amount} country's rate from one of the conversion rate tables. See “Example 1: Converting from Deutsche Marks to Euros” on page 818. If you are converting from

<table>
<thead>
<tr>
<th>Currency</th>
<th>Currency code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Danish krone</td>
<td>DKK</td>
</tr>
<tr>
<td>Deutsche mark</td>
<td>DEM</td>
</tr>
<tr>
<td>Dutch guilder</td>
<td>NLG</td>
</tr>
<tr>
<td>Euro</td>
<td>EUR</td>
</tr>
<tr>
<td>Finnish markka</td>
<td>FIM</td>
</tr>
<tr>
<td>French franc</td>
<td>FRF</td>
</tr>
<tr>
<td>Greek drachma</td>
<td>GRD</td>
</tr>
<tr>
<td>Hungarian forint</td>
<td>HUF</td>
</tr>
<tr>
<td>Irish pound</td>
<td>IEP</td>
</tr>
<tr>
<td>Italian lira</td>
<td>ITL</td>
</tr>
<tr>
<td>Luxembourg franc</td>
<td>LUF</td>
</tr>
<tr>
<td>Norwegian krone</td>
<td>NOK</td>
</tr>
<tr>
<td>Polish zloty</td>
<td>PLZ</td>
</tr>
<tr>
<td>Portuguese escudo</td>
<td>PTE</td>
</tr>
<tr>
<td>Romanian leu</td>
<td>ROL</td>
</tr>
<tr>
<td>Russian ruble</td>
<td>RUR</td>
</tr>
<tr>
<td>Slovenian tolar</td>
<td>SIT</td>
</tr>
<tr>
<td>Spanish peseta</td>
<td>ESP</td>
</tr>
<tr>
<td>Swedish krona</td>
<td>SEK</td>
</tr>
<tr>
<td>Swiss franc</td>
<td>CHF</td>
</tr>
<tr>
<td>Turkish lira</td>
<td>TRL</td>
</tr>
</tbody>
</table>
euros to a country's currency, SAS multiplies by that country's rate from one of the conversion rate tables. See “Example 2: Converting from Euros to Deutsche Marks” on page 818. If you are converting one country's currency to another country's currency, SAS first converts the from-currency-amount to euros. SAS stores the intermediate value in as much precision as your operating environment allows, and does not round the value. SAS then converts the amount in euros to an amount in the currency that you are converting to. See “Example 3: Converting from French Francs to Deutsche Marks” on page 818.

Examples

**Example 1: Converting from Deutsche Marks to Euros**
The following example converts one Deutsche mark to an equivalent amount of euros.

```sas
data _null_;  
   amount=eurocurr(50,'dem','eur');  
   put amount=;  
run;
```

The value in the SAS log is: **amount=25.56459406**.

**Example 2: Converting from Euros to Deutsche Marks**
The following example converts one euro to an equivalent amount of Deutsche marks.

```sas
data _null_;  
   amount=eurocurr(25,'eur','dem');  
   put amount=;  
run;
```

The value in the SAS log is: **amount=48.89575**.

**Example 3: Converting from French Francs to Deutsche Marks**
The following example converts 50 French francs to an equivalent amount of Deutsche marks.

```sas
data _null_;  
x=50;  
   amount=eurocurr(x,'frf','dem');  
   put amount=;  
run;
```

The value in the SAS log is: **amount=14.908218069**.

**Example 4: Converting Currency When One Variable Is Blank**
The following example converts 50 euros to Deutsche marks.

```sas
data _null_;  
x=50;  
   amount=eurocurr(x,' ','dem');  
   put amount=;  
run;
```

The value in the SAS log is: **amount=97.7915**.
## Appendix 2

**Time Zone IDs and Time Zone Names**

---

### Area: Africa

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa/Abidjan</td>
<td>GMT</td>
<td>CI</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Accra</td>
<td>GMT</td>
<td>GH</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Bamako</td>
<td>GMT</td>
<td>ML</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Banjul</td>
<td>GMT</td>
<td>GM</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Bissau</td>
<td>GMT</td>
<td>GW</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Conakry</td>
<td>GMT</td>
<td>GN</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Dakar</td>
<td>GMT</td>
<td>SN</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Freetown</td>
<td>GMT</td>
<td>SL</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Lome</td>
<td>GMT</td>
<td>TG</td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Africa/Monrovia</td>
<td>GMT LR</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Nouakchott</td>
<td>GMT MR</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Ouagadougou</td>
<td>GMT BF</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Sao_Tome</td>
<td>GMT ST</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>00:00</td>
</tr>
<tr>
<td>Africa/Timbuktu</td>
<td>GMT ML</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0.00</td>
</tr>
<tr>
<td>Africa/Casablanca</td>
<td>WET WEST MA</td>
<td></td>
<td>Western European Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>WET WEST MA</td>
<td></td>
<td>Western European Summer Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/El_Aaiun</td>
<td>WET WEST EH</td>
<td></td>
<td>Western European Time</td>
<td>00:00</td>
</tr>
<tr>
<td></td>
<td>WET WEST EH</td>
<td></td>
<td>Western European Summer Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Algiers</td>
<td>CET DZ</td>
<td></td>
<td>Central European Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Algiers</td>
<td>CET ES</td>
<td></td>
<td>Central European Time</td>
<td>01:00</td>
</tr>
<tr>
<td></td>
<td>CEST ES</td>
<td></td>
<td>Central European Summer Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Tunis</td>
<td>CET TN</td>
<td></td>
<td>Central European Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Bangui</td>
<td>WAT CF</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Brazzaville</td>
<td>WAT CG</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Douala</td>
<td>WAT CM</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Kinshasa</td>
<td>WAT CD</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Lagos</td>
<td>WAT NG</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Libreville</td>
<td>WAT GA</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Luanda</td>
<td>WAT AO</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Malabo</td>
<td>WAT GQ</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Ndjamena</td>
<td>WAT TD</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Niamey</td>
<td>WAT NE</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Porto-Novio</td>
<td>WAT BJ</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td>Africa/Windhoek</td>
<td>WAT NA</td>
<td></td>
<td>West Africa Time</td>
<td>01:00</td>
</tr>
<tr>
<td></td>
<td>WAST NA</td>
<td></td>
<td>West Africa Summer Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Blantyre</td>
<td>CAT MW</td>
<td></td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Africa/Bujumbura</td>
<td>CAT</td>
<td>BI</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Gaborone</td>
<td>CAT</td>
<td>BW</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Harare</td>
<td>CAT</td>
<td>ZW</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Kigali</td>
<td>CAT</td>
<td>RW</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Lubumbashi</td>
<td>CAT</td>
<td>CD</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Lusaka</td>
<td>CAT</td>
<td>ZM</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Maputo</td>
<td>CAT</td>
<td>MZ</td>
<td>Central Africa Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Cairo</td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Time</td>
<td>02:00 2:00</td>
</tr>
<tr>
<td>Africa/Maputo</td>
<td>CAT</td>
<td>MZ</td>
<td>Central African Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Maputo</td>
<td>EET</td>
<td>LY</td>
<td>Eastern European Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Johannesburg</td>
<td>SAST</td>
<td>ZA</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Maseru</td>
<td>SAST</td>
<td>LS</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Mbabane</td>
<td>SAST</td>
<td>SZ</td>
<td>South Africa Standard Time</td>
<td>02:00</td>
</tr>
<tr>
<td>Africa/Addis_Ababa</td>
<td>EAT</td>
<td>ET</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Asmara</td>
<td>EAT</td>
<td>ER</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Asmera</td>
<td>EAT</td>
<td>ER</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Dar_es_Salaam</td>
<td>EAT</td>
<td>TZ</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Djibouti</td>
<td>EAT</td>
<td>DJ</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Juba</td>
<td>EAT</td>
<td>SS</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Kampala</td>
<td>EAT</td>
<td>UG</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Khartoum</td>
<td>EAT</td>
<td>SD</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Mogadishu</td>
<td>EAT</td>
<td>SO</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
<tr>
<td>Africa/Nairobi</td>
<td>EAT</td>
<td>KE</td>
<td>East Africa Time</td>
<td>03:00</td>
</tr>
</tbody>
</table>
### Area: America–North, Central, and South

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>America/Adak</td>
<td>HAST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>America/Adak</td>
<td>HADT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Anchorage</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Anchorage</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Anguilla</td>
<td>AST</td>
<td>AI</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Antigua</td>
<td>AST</td>
<td>AG</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Araguaina</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Buenos Aires</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Catamarca</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/ComodRivadavia</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Cordoba</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Jujuy</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/La_Rioja</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Mendoza</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Rio_Gallegos</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Salta</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/San_Juan</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/San_Luis</td>
<td>ART</td>
<td>AR</td>
<td>Western Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Tucuman</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Argentina/Ushuaia</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Aruba</td>
<td>AST</td>
<td>AW</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Asuncion</td>
<td>PYT</td>
<td>PY</td>
<td>Paraguay Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Asuncion</td>
<td>PYST</td>
<td>PY</td>
<td>Paraguay Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Atikokan</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Atka</td>
<td>HAST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Atka</td>
<td>HADT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Bahia</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Bahia_Banderas</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Bahia_Banderas</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Barbados</td>
<td>AST</td>
<td>BB</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Belem</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Belize</td>
<td>CST</td>
<td>BZ</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Blanc-Sablon</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Boa_Vista</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Bogota</td>
<td>COT</td>
<td>CO</td>
<td>Colombia Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Boise</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Boise</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Buenos_Aires</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cambridge_Bay</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Cambridge_Bay</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Campo_Grande</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Campo_Grande</td>
<td>AMST</td>
<td>BR</td>
<td>Amazon Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cancun</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Cancun</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Caracas</td>
<td>VET</td>
<td>VE</td>
<td>Venezuela Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>America/Catamarca</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cayenne</td>
<td>GFT</td>
<td>GF</td>
<td>French Guiana Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Cayman</td>
<td>EST</td>
<td>KY</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Chicago</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Chicago</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Chihuahua</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Chihuahua</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Coral_Harbour</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Cordoba</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Costa Rica</td>
<td>CST</td>
<td>CR</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Creston</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Cuiaba</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Cuiaba</td>
<td>AMST</td>
<td>BR</td>
<td>Amazon Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Curacao</td>
<td>AST</td>
<td>CW</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Danmarkshavn</td>
<td>GMT</td>
<td>GL</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>America/Dawson</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Dawson</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Dawson_Creek</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Denver</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Denver</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Detroit</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Detroit</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Dominica</td>
<td>AST</td>
<td>DM</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Edmonton</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Edmonton</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Eirunepe</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/El_Salvador</td>
<td>CST</td>
<td>SV</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Ensenada</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Ensenada</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Fort_Wayne</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Fort_Wayne</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Fortaleza</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Glace_Bay</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Glace_Bay</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Godthab</td>
<td>WGT</td>
<td>GL</td>
<td>West Greenland Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Godthab</td>
<td>WGST</td>
<td>GL</td>
<td>West Greenland Summer Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Goose_Bay</td>
<td>AST CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Goose_Bay</td>
<td>ADT CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
<td></td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EST TC</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Grand_Turk</td>
<td>EDT TC</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Grenada</td>
<td>AST GD</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Guadeloupe</td>
<td>AST GP</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Guadeloupe</td>
<td>AST GP</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Guatemala</td>
<td>CST GT</td>
<td>Central Standard Time</td>
<td>-6:00</td>
<td></td>
</tr>
<tr>
<td>America/Guayaquil</td>
<td>ECT EC</td>
<td>Ecuador Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Guyana</td>
<td>GYT GY</td>
<td>Guyana Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Halifax</td>
<td>EST CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Halifax</td>
<td>EDT CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
<td></td>
</tr>
<tr>
<td>America/Havana</td>
<td>CST CU</td>
<td>Cuba Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Havana</td>
<td>CDT CU</td>
<td>Cuba Daylight Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Hermosillo</td>
<td>MST MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EST US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Indianapolis</td>
<td>EDT US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CST US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Knox</td>
<td>CDT US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EST US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Marengo</td>
<td>EDT US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EST US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Petersburg</td>
<td>EDT US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CST US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Tell_City</td>
<td>CDT US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EST US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Vevay</td>
<td>EDT US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Vincennes</td>
<td>EST US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
<td></td>
</tr>
<tr>
<td>America/Indiana/Vincennes</td>
<td>EDT US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
<td></td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Indiana/Winamac</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indiana/Winamac</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Indianapolis</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Indianapolis</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Inuvik</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Inuvik</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Iqaluit</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Iqaluit</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Jamaica</td>
<td>EST</td>
<td>JM</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Jujuy</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Juneau</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Juneau</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Kentucky/Louisville</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Kentucky/Louisville</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Kentucky/Monticello</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Kentucky/Monticello</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Knox_IN</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Knox_IN</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Kralendijk</td>
<td>AST</td>
<td>BQ</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/La_Paz</td>
<td>BOT</td>
<td>BO</td>
<td>Bolivia Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Lima</td>
<td>PET</td>
<td>PE</td>
<td>Peru Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Los_Angeles</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Los_Angeles</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Louisville</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Louisville</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Lower_Princes</td>
<td>AST</td>
<td>SX</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Maceio</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Managua</td>
<td>CST</td>
<td>NI</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Manaus</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Marigot</td>
<td>AST</td>
<td>MF</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Martinique</td>
<td>AST</td>
<td>MQ</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Matamoros</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Matamoros</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Mazatlan</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Mazatlan</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Mendoza</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Menominee</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Menominee</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Merida</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Merida</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Metlakatla</td>
<td>MEST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Mexico_City</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Mexico_City</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Miquelon</td>
<td>PMST</td>
<td>PM</td>
<td>Saint Pierre and Miquelon Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Miquelon</td>
<td>PMDT</td>
<td>PM</td>
<td>Saint Pierre and Miquelon Daylight Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/Moncton</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Moncton</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Monterrey</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Monterrey</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Montevideo</td>
<td>UYT</td>
<td>UY</td>
<td>Uruguay Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Montevideo</td>
<td>UYST</td>
<td>UY</td>
<td>Uruguay Summer Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/Montserrat</td>
<td>AST</td>
<td>MS</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Nassau</td>
<td>EST</td>
<td>BS</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Nassau</td>
<td>EDT</td>
<td>BS</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/New_York</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/New_York</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Nipigon</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Nipigon</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Nome</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Nome</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Noronha</td>
<td>FNT</td>
<td>BR</td>
<td>Fernando de Noronha Standard Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/North_Dakota/Beulah</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/North_Dakota/Beulah</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/North_Dakota/Center</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/North_Dakota/Center</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/North_Dakota/New_Salem</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/North_Dakota/New_Salem</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Ojinaga</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Ojinaga</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Panama</td>
<td>EST</td>
<td>PA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Pangnirtung</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Pangnirtung</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Paramaribo</td>
<td>SRT</td>
<td>SR</td>
<td>Suriname Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Phoenix</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Port-au-Prince</td>
<td>EST</td>
<td>HT</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Port-au-Prince</td>
<td>EDT</td>
<td>HT</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Port_of_Spain</td>
<td>AST</td>
<td>TT</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Porto_Acre</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Porto_Velho</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Puerto_Rico</td>
<td>AST</td>
<td>PR</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Rainy_River</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Rainy_River</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>--------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/Rankin_Inlet</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Rankin_Inlet</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Recife</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Regina</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Resolute</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Resolute</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Rio_Branco</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Rosario</td>
<td>ART</td>
<td>AR</td>
<td>Argentina Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Santa_Isabel</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Santa_Isabel</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Santarem</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Santiago</td>
<td>CLT</td>
<td>CL</td>
<td>Chile Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Santiago</td>
<td>CLST</td>
<td>CL</td>
<td>Chile Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Santo_Domingo</td>
<td>AST</td>
<td>DO</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Sao_Paulo</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Sao_Paulo</td>
<td>BRST</td>
<td>BR</td>
<td>Brasilia Summer Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>America/Scoresbysund</td>
<td>EGT</td>
<td>GL</td>
<td>East Greenland Standard Time</td>
<td>-1:00</td>
</tr>
<tr>
<td>America/Scoresbysund</td>
<td>EGST</td>
<td>GL</td>
<td>East Greenland Summer Time</td>
<td>0:00</td>
</tr>
<tr>
<td>America/Shiprock</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Shiprock</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Sitka</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Sitka</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/St_Barthelemy</td>
<td>AST</td>
<td>BL</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/St_Johns</td>
<td>NST</td>
<td>CA</td>
<td>Newfoundland Standard Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>America/St_Johns</td>
<td>NDT</td>
<td>CA</td>
<td>Newfoundland Daylight Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>America/St_Kitts</td>
<td>AST</td>
<td>KN</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/St_Lucia</td>
<td>AST</td>
<td>LC</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/St_Thomas</td>
<td>AST</td>
<td>VI</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>America/St_Vincent</td>
<td>AST</td>
<td>VC</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Swift_Current</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Tegucigalpa</td>
<td>CST</td>
<td>HN</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Thule</td>
<td>AST</td>
<td>GL</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Thule</td>
<td>ADT</td>
<td>GL</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>America/Thunder_Bay</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Thunder_Bay</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Tijuana</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Tijuana</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Toronto</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Toronto</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Tortola</td>
<td>AST</td>
<td>VG</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Vancouver</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Vancouver</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Virgin</td>
<td>AST</td>
<td>TT</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>America/Whitehorse</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Whitehorse</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Winnipeg</td>
<td>CST</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>America/Winnipeg</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>America/Yakutat</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>America/Yakutat</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>America/Yellowknife</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>America/Yellowknife</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
</tbody>
</table>
### Area: Antarctica

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antarctica/Casey</td>
<td>WST</td>
<td>AQ</td>
<td>Australian Western Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Antarctica/Davis</td>
<td>DAVT</td>
<td>AQ</td>
<td>Davis Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Antarctica/DumontDUrville</td>
<td>DDUT</td>
<td>AQ</td>
<td>Dumont-d'Urville Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Antarctica/Macquarie</td>
<td>MIST</td>
<td>AU</td>
<td>Macquarie Island Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Antarctica/Mawson</td>
<td>MAWT</td>
<td>AQ</td>
<td>Mawson Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Antarctica/McMurdo</td>
<td>NZST</td>
<td>AQ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Antarctica/McMurdo</td>
<td>NZDT</td>
<td>AQ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Antarctica/Palmer</td>
<td>CLT</td>
<td>AQ</td>
<td>Chile Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Antarctica/Palmer</td>
<td>CLST</td>
<td>AQ</td>
<td>Chile Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Antarctica/Rothera</td>
<td>ROTT</td>
<td>AQ</td>
<td>Rothera Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Antarctica/South_Pole</td>
<td>NZST</td>
<td>NZ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Antarctica/South_Pole</td>
<td>NZDT</td>
<td>NZ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Antarctica/Syowa</td>
<td>SYOT</td>
<td>AQ</td>
<td>Syowa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Antarctica/Vostok</td>
<td>VOST</td>
<td>AQ</td>
<td>Vostok Time</td>
<td>6:00</td>
</tr>
</tbody>
</table>

### Area: Asia

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia/Aden</td>
<td>AST</td>
<td>YE</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Almaty</td>
<td>ALMT</td>
<td>KZ</td>
<td>East Kazakhstan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Amman</td>
<td>EET</td>
<td>JO</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Amman</td>
<td>EEST</td>
<td>JO</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Anadyr</td>
<td>ANAT</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Asia/Aqtau</td>
<td>AQTT</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Asia/Aqtobe</td>
<td>AQTT</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Ashgabat</td>
<td>TMT</td>
<td>TM</td>
<td>Turkmenistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Ashkhabad</td>
<td>TMT</td>
<td>TM</td>
<td>Turkmenistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Baghdad</td>
<td>AST</td>
<td>IQ</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Bahrain</td>
<td>AST</td>
<td>BH</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Baku</td>
<td>AZT</td>
<td>AZ</td>
<td>Azerbaijan Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Baku</td>
<td>AZST</td>
<td>AZ</td>
<td>Azerbaijan Summer Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Bangkok</td>
<td>ICT</td>
<td>TH</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Beijing</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Beirut</td>
<td>EET</td>
<td>LB</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Beirut</td>
<td>EEST</td>
<td>LB</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Bishkek</td>
<td>KGT</td>
<td>KG</td>
<td>Kyrgyzstan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Brunei</td>
<td>BNT</td>
<td>BN</td>
<td>Brunei Darussalam Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Calcutta</td>
<td>IST</td>
<td>IN</td>
<td>India Standard Time</td>
<td>5:30</td>
</tr>
<tr>
<td>Asia/Choibalsan</td>
<td>CHOT</td>
<td>MN</td>
<td>Choibalsan Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Chongqing</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Chungking</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Colombo</td>
<td>IST</td>
<td>LK</td>
<td>India Standard Time</td>
<td>5:30</td>
</tr>
<tr>
<td>Asia/Dacca</td>
<td>BDT</td>
<td>BD</td>
<td>Bangladesh Standard Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Damascus</td>
<td>EET</td>
<td>SY</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Damascus</td>
<td>EEST</td>
<td>SY</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Dhaka</td>
<td>BDT</td>
<td>BD</td>
<td>Bangladesh Standard Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Dili</td>
<td>TLT</td>
<td>TL</td>
<td>East Timor Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Dubai</td>
<td>GST</td>
<td>AE</td>
<td>Gulf Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Dushanbe</td>
<td>TJT</td>
<td>TJ</td>
<td>Tajikistan Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Gaza</td>
<td>EET</td>
<td>PS</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Gaza</td>
<td>EEST</td>
<td>PS</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Harbin</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Asia/Hebron</td>
<td>EET</td>
<td>PS</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Hebron</td>
<td>EEST</td>
<td>PS</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Ho_Chi_Minh</td>
<td>ICT</td>
<td>VN</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Hong_Kong</td>
<td>HKT</td>
<td>HK</td>
<td>Hong Kong Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Hovd</td>
<td>HOVT</td>
<td>MN</td>
<td>Hovd Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Irkutsk</td>
<td>IRKT</td>
<td>RU</td>
<td>Irkutsk Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Istanbul</td>
<td>EET</td>
<td>TR</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Istanbul</td>
<td>EEST</td>
<td>TR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Jakarta</td>
<td>WIB</td>
<td>ID</td>
<td>Western Indonesia Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Jayapura</td>
<td>WIT</td>
<td>ID</td>
<td>Eastern Indonesia Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Jerusalem</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Jerusalem</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Kabul</td>
<td>AFT</td>
<td>AF</td>
<td>Afghanistan Time</td>
<td>4:30</td>
</tr>
<tr>
<td>Asia/Amman</td>
<td>PETT</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Asia/Karachi</td>
<td>PKT</td>
<td>PK</td>
<td>Pakistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Kashgar</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kathmandu</td>
<td>NPT</td>
<td>NP</td>
<td>Nepal Time</td>
<td>5:45</td>
</tr>
<tr>
<td>Asia/Katmandu</td>
<td>NPT</td>
<td>NP</td>
<td>Nepal Time</td>
<td>5:45</td>
</tr>
<tr>
<td>Asia/Khandyga</td>
<td>YAKT</td>
<td>RU</td>
<td>Yakutsk Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Asia/Kolkata</td>
<td>IST</td>
<td>IN</td>
<td>India Standard Time</td>
<td>5:30</td>
</tr>
<tr>
<td>Asia/Krasnoyarsk</td>
<td>KRAT</td>
<td>RU</td>
<td>Krasnoyarsk Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kuala_Lumpur</td>
<td>MYT</td>
<td>MY</td>
<td>Malaysia Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kuching</td>
<td>MYT</td>
<td>MY</td>
<td>Malaysia Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Kuwait</td>
<td>AST</td>
<td>KW</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Macao</td>
<td>CST</td>
<td>MO</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Macau</td>
<td>CST</td>
<td>MO</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Magadan</td>
<td>MAGT</td>
<td>RU</td>
<td>Magadan Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Asia/Makassar</td>
<td>WITA</td>
<td>ID</td>
<td>Central Indonesia Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Asia/Manila</td>
<td>PHT</td>
<td>PH</td>
<td>Philippine Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Muscat</td>
<td>GST</td>
<td>OM</td>
<td>Gulf Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Nicosia</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Asia/Nicosia</td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Novokuznetsk</td>
<td>NOVT</td>
<td>RU</td>
<td>Novosibirsk Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Novosibirsk</td>
<td>NOVT</td>
<td>RU</td>
<td>Novosibirsk Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Omsk</td>
<td>OMST</td>
<td>RU</td>
<td>Omsk Standard Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Oral</td>
<td>ORAT</td>
<td>KZ</td>
<td>West Kazakhstan Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Osaka</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Phnom_Penh</td>
<td>ICT</td>
<td>KH</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Pontianak</td>
<td>WIB</td>
<td>ID</td>
<td>Western Indonesia Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Pyongyang</td>
<td>KST</td>
<td>KP</td>
<td>Korean Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Qatar</td>
<td>AST</td>
<td>QA</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Qyzylorda</td>
<td>QYZT</td>
<td>KZ</td>
<td>East Kazakhstan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Rangoon</td>
<td>MMT</td>
<td>MM</td>
<td>Myanmar Time</td>
<td>6:30</td>
</tr>
<tr>
<td>Asia/Riyadh</td>
<td>AST</td>
<td>SA</td>
<td>Arabian Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Saigon</td>
<td>ICT</td>
<td>VN</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Sakhalin</td>
<td>SAKT</td>
<td>RU</td>
<td>Sakhalin Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Samarkand</td>
<td>UZT</td>
<td>UZ</td>
<td>Uzbekistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Sapporo</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Seoul</td>
<td>KST</td>
<td>KR</td>
<td>Korean Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Shanghai</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Singapore</td>
<td>SGT</td>
<td>SG</td>
<td>Singapore Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Taipei</td>
<td>CST</td>
<td>TW</td>
<td>Taipei Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Tashkent</td>
<td>UZT</td>
<td>UZ</td>
<td>Uzbekistan Standard Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Asia/Tbilisi</td>
<td>GET</td>
<td>GE</td>
<td>Georgia Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Asia/Tehran</td>
<td>IRST</td>
<td>IR</td>
<td>Iran Standard Time</td>
<td>3:30</td>
</tr>
<tr>
<td>Asia/Tel_Aviv</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Asia/Tel_Aviv</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Asia/Thimbu</td>
<td>BTT</td>
<td>BT</td>
<td>Bhutan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Thimphu</td>
<td>BTT</td>
<td>BT</td>
<td>Bhutan Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Tokyo</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Asia/Ujung_Pandang</td>
<td>WITA</td>
<td>ID</td>
<td>Central Indonesia Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Ulaanbaatar</td>
<td>ULAT</td>
<td>MN</td>
<td>Ulan Bator Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Ulan_Bator</td>
<td>ULAT</td>
<td>MN</td>
<td>Ulan Bator Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Urumqi</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Asia/Ust-Nera</td>
<td>VLAT</td>
<td>RU</td>
<td>Vladivostok Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Vientiane</td>
<td>ICT</td>
<td>LA</td>
<td>Indochina Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Asia/Vladivostok</td>
<td>VLAT</td>
<td>RU</td>
<td>Vladivostok Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Asia/Yakutsk</td>
<td>YAKT</td>
<td>RU</td>
<td>Yakutsk Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Asia/Yekaterinburg</td>
<td>YEKT</td>
<td>RU</td>
<td>Yekaterinburg Standard Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Asia/Yerevan</td>
<td>AMT</td>
<td>AM</td>
<td>Armenia Standard Time</td>
<td>4:00</td>
</tr>
</tbody>
</table>

**Area: Atlantic**

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlantic/Azores</td>
<td>AZOT</td>
<td>PT</td>
<td>Azores Standard Time</td>
<td>-1:00</td>
</tr>
<tr>
<td>Atlantic/Azores</td>
<td>AZOST</td>
<td>PT</td>
<td>Azores Summer Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Bermuda</td>
<td>AST</td>
<td>BM</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Atlantic/Bermuda</td>
<td>ADT</td>
<td>BM</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Atlantic/Canary</td>
<td>WET</td>
<td>ES</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Canary</td>
<td>WEST</td>
<td>ES</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Cape_Verde</td>
<td>CVT</td>
<td>CV</td>
<td>Cape Verde Standard Time</td>
<td>-1:00</td>
</tr>
<tr>
<td>Atlantic/Faeroe</td>
<td>WET</td>
<td>FO</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Faeroe</td>
<td>WEST</td>
<td>FO</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
</tbody>
</table>
## Time Zone Information

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlantic/Faroe</td>
<td>WET</td>
<td>FO</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Faroe</td>
<td>WEST</td>
<td>FO</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CET</td>
<td>NO</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Jan_Mayen</td>
<td>CEST</td>
<td>NO</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Madeira</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Atlantic/Reyjavik</td>
<td>GMT</td>
<td>IS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/South_Georgia</td>
<td>GST</td>
<td>GS</td>
<td>South Georgia Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Atlantic/St_Helena</td>
<td>GMT</td>
<td>SH</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Atlantic/Stanley</td>
<td>FKST</td>
<td>FK</td>
<td>Falkland Islands Standard Time</td>
<td>-3:00</td>
</tr>
</tbody>
</table>

## Area: Australia

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia/ACT</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/ACT</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Adelaide</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Brisbane</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Broken_Hill</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Canberra</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Currie</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Darwin</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Australia/Eucla</td>
<td>CWST</td>
<td>AU</td>
<td>Australian Central Western Standard Time</td>
<td>8:45</td>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Hobart</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>LHST</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/LHI</td>
<td>LHDT</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Lindeman</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>LHST</td>
<td>AU</td>
<td>Lord Howe Standard Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Lord_Howe</td>
<td>LHDT</td>
<td>AU</td>
<td>Lord Howe Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Melbourne</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/NSW</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/North</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Perth</td>
<td>WST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Australia/Queensland</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/South</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/South</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Sydney</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Tasmania</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>EST</td>
<td>AU</td>
<td>Australian Eastern Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Australia/Victoria</td>
<td>EDT</td>
<td>AU</td>
<td>Australian Eastern Daylight Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Australia/West</td>
<td>WST</td>
<td>AU</td>
<td>Australian Western Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>CST</td>
<td>AU</td>
<td>Australian Central Standard Time</td>
<td>9:30</td>
</tr>
<tr>
<td>Australia/Yancowinna</td>
<td>CDT</td>
<td>AU</td>
<td>Australian Central Daylight Time</td>
<td>10:30</td>
</tr>
</tbody>
</table>
## Area: Miscellaneous

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arctic/Longyearbyen</td>
<td>CET</td>
<td>SJ</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Arctic/Longyearbyen</td>
<td>CEST</td>
<td>SJ</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Brazil/Acre</td>
<td>ACT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Brazil/DeNoronha</td>
<td>FNT</td>
<td>BR</td>
<td>Fernando de Noronha Standard Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Brazil/East</td>
<td>BRT</td>
<td>BR</td>
<td>Brasilia Standard Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Brazil/East</td>
<td>BRST</td>
<td>BR</td>
<td>Brasilia Summer Time</td>
<td>-2:00</td>
</tr>
<tr>
<td>Brazil/West</td>
<td>AMT</td>
<td>BR</td>
<td>Amazon Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>CST6CDT</td>
<td>CST</td>
<td></td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>CST6CDT</td>
<td>CDT</td>
<td></td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Canada/Atlantic</td>
<td>AST</td>
<td>CA</td>
<td>Atlantic Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Canada/Atlantic</td>
<td>ADT</td>
<td>CA</td>
<td>Atlantic Daylight Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Canada/Central</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Central</td>
<td>CDT</td>
<td>CA</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Canada/East-Saskatchewan</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Eastern</td>
<td>EST</td>
<td>CA</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Canada/Eastern</td>
<td>EDT</td>
<td>CA</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Canada/Mountain</td>
<td>MST</td>
<td>CA</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Canada/Mountain</td>
<td>MDT</td>
<td>CA</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Newfoundland</td>
<td>NST</td>
<td>CA</td>
<td>Newfoundland Standard Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>Canada/Newfoundland</td>
<td>NDT</td>
<td>CA</td>
<td>Newfoundland Daylight Time</td>
<td>-1:27</td>
</tr>
<tr>
<td>Canada/Pacific</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>Canada/Pacific</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Canada/Saskatchewan</td>
<td>CST</td>
<td>CA</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Canada/Yukon</td>
<td>PST</td>
<td>CA</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>Canada/Yukon</td>
<td>PDT</td>
<td>CA</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Chile/Continental</td>
<td>CLT</td>
<td>CL</td>
<td>Chile Standard Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Chile/Continental</td>
<td>CLST</td>
<td>CL</td>
<td>Chile Summer Time</td>
<td>-3:00</td>
</tr>
<tr>
<td>Chile/EasterIsland</td>
<td>EAST</td>
<td>CL</td>
<td>Easter Island Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Chile/EasterIsland</td>
<td>EASST</td>
<td>CL</td>
<td>Easter Island Summer Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Cuba</td>
<td>CST</td>
<td>CU</td>
<td>Cuba Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Cuba</td>
<td>CDT</td>
<td>CU</td>
<td>Cuba Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>EST5EDT</td>
<td>EST</td>
<td></td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>EST5EDT</td>
<td>EDT</td>
<td></td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Egypt</td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Standard Time</td>
<td>+3:00</td>
</tr>
<tr>
<td>Egypt</td>
<td>EET</td>
<td>EG</td>
<td>Eastern European Summer Time</td>
<td>+2:00</td>
</tr>
<tr>
<td>Eire</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Eire</td>
<td>IST</td>
<td>IE</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Etc/GMT</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/GMT+0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/GMT+1</td>
<td>GMT-01</td>
<td></td>
<td>1 hours behind GMT</td>
<td>-1:00</td>
</tr>
<tr>
<td>Etc/GMT+10</td>
<td>GMT-10</td>
<td></td>
<td>10 hours behind GMT</td>
<td>-10:00</td>
</tr>
<tr>
<td>Etc/GMT+11</td>
<td>GMT-11</td>
<td></td>
<td>11 hours behind GMT</td>
<td>-11:00</td>
</tr>
<tr>
<td>Etc/GMT+12</td>
<td>GMT-12</td>
<td></td>
<td>12 hours behind GMT</td>
<td>-12:00</td>
</tr>
<tr>
<td>Etc/GMT+2</td>
<td>GMT-02</td>
<td></td>
<td>2 hours behind GMT</td>
<td>-2:00</td>
</tr>
<tr>
<td>Etc/GMT+3</td>
<td>GMT-03</td>
<td></td>
<td>3 hours behind GMT</td>
<td>-3:00</td>
</tr>
<tr>
<td>Etc/GMT+4</td>
<td>GMT-04</td>
<td></td>
<td>4 hours behind GMT</td>
<td>-4:00</td>
</tr>
<tr>
<td>Etc/GMT+5</td>
<td>GMT-05</td>
<td></td>
<td>5 hours behind GMT</td>
<td>-5:00</td>
</tr>
<tr>
<td>Etc/GMT+6</td>
<td>GMT-06</td>
<td></td>
<td>6 hours behind GMT</td>
<td>-6:00</td>
</tr>
<tr>
<td>Etc/GMT+7</td>
<td>GMT-07</td>
<td></td>
<td>7 hours behind GMT</td>
<td>-7:00</td>
</tr>
<tr>
<td>Etc/GMT+8</td>
<td>GMT-08</td>
<td></td>
<td>8 hours behind GMT</td>
<td>-8:00</td>
</tr>
<tr>
<td>Etc/GMT+9</td>
<td>GMT-09</td>
<td></td>
<td>9 hours behind GMT</td>
<td>-9:00</td>
</tr>
<tr>
<td>Etc/GMT-0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/GMT-1</td>
<td>GMT+01</td>
<td></td>
<td>1 hours ahead of GMT</td>
<td>1:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Etc/GMT-10</td>
<td>GMT+10</td>
<td></td>
<td>10 hours ahead of GMT</td>
<td>10:00</td>
</tr>
<tr>
<td>Etc/GMT-11</td>
<td>GMT+11</td>
<td></td>
<td>11 hours ahead of GMT</td>
<td>11:00</td>
</tr>
<tr>
<td>Etc/GMT-12</td>
<td>GMT+12</td>
<td></td>
<td>12 hours ahead of GMT</td>
<td>12:00</td>
</tr>
<tr>
<td>Etc/GMT-13</td>
<td>GMT+13</td>
<td></td>
<td>13 hours ahead of GMT</td>
<td>13:00</td>
</tr>
<tr>
<td>Etc/GMT-14</td>
<td>GMT+14</td>
<td></td>
<td>14 hours ahead of GMT</td>
<td>14:00</td>
</tr>
<tr>
<td>Etc/GMT-2</td>
<td>GMT+02</td>
<td></td>
<td>2 hours ahead of GMT</td>
<td>2:00</td>
</tr>
<tr>
<td>Etc/GMT-3</td>
<td>GMT+03</td>
<td></td>
<td>3 hours ahead of GMT</td>
<td>3:00</td>
</tr>
<tr>
<td>Etc/GMT-4</td>
<td>GMT+04</td>
<td></td>
<td>4 hours ahead of GMT</td>
<td>4:00</td>
</tr>
<tr>
<td>Etc/GMT-5</td>
<td>GMT+05</td>
<td></td>
<td>5 hours ahead of GMT</td>
<td>5:00</td>
</tr>
<tr>
<td>Etc/GMT-6</td>
<td>GMT+06</td>
<td></td>
<td>6 hours ahead of GMT</td>
<td>6:00</td>
</tr>
<tr>
<td>Etc/GMT-7</td>
<td>GMT+07</td>
<td></td>
<td>7 hours ahead of GMT</td>
<td>7:00</td>
</tr>
<tr>
<td>Etc/GMT-8</td>
<td>GMT+08</td>
<td></td>
<td>8 hours ahead of GMT</td>
<td>8:00</td>
</tr>
<tr>
<td>Etc/GMT-9</td>
<td>GMT+09</td>
<td></td>
<td>9 hours ahead of GMT</td>
<td>9:00</td>
</tr>
<tr>
<td>Etc/GMT0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/Greenwich</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/UCT</td>
<td>UCT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/UTC</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/Universal</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Etc/Zulu</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GB</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GB</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>1:00</td>
</tr>
<tr>
<td>GB-Eire</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GB-Eire</td>
<td>BST</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>1:00</td>
</tr>
<tr>
<td>GMT</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GMT+0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GMT-0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>GMT0</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Greenwich</td>
<td>GMT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>----------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Hongkong</td>
<td>HKT</td>
<td>HK</td>
<td>Hong Kong Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Iceland</td>
<td>GMT</td>
<td>IS</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Indian/Antananarivo</td>
<td>EAT</td>
<td>MG</td>
<td>East Africa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Indian/Chagos</td>
<td>IOT</td>
<td>IO</td>
<td>Indian Ocean Time</td>
<td>6:00</td>
</tr>
<tr>
<td>Indian/Christmas</td>
<td>CXT</td>
<td>CX</td>
<td>Christmas Island Time</td>
<td>7:00</td>
</tr>
<tr>
<td>Indian/Cocos</td>
<td>CCT</td>
<td>CC</td>
<td>Cocos Islands Time</td>
<td>6:30</td>
</tr>
<tr>
<td>Indian/Comoro</td>
<td>EAT</td>
<td>KM</td>
<td>East Africa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Indian/Kerguelen</td>
<td>TFT</td>
<td>TF</td>
<td>French Southern and Antarctic Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Indian/Mahe</td>
<td>SCT</td>
<td>SC</td>
<td>Seychelles Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Indian/Maldives</td>
<td>MVT</td>
<td>MV</td>
<td>Maldives Time</td>
<td>5:00</td>
</tr>
<tr>
<td>Indian/Mauritius</td>
<td>MUT</td>
<td>MU</td>
<td>Mauritius Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Indian/Mayotte</td>
<td>EAT</td>
<td>YT</td>
<td>East Africa Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Indian/Reunion</td>
<td>RET</td>
<td>RE</td>
<td>Reunion Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Iran</td>
<td>IRST</td>
<td>IR</td>
<td>Iran Standard Time</td>
<td>3:30</td>
</tr>
<tr>
<td>Israel</td>
<td>IST</td>
<td>IL</td>
<td>Israel Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Israel</td>
<td>IDT</td>
<td>IL</td>
<td>Israel Daylight Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Jamaica</td>
<td>EST</td>
<td>JM</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Japan</td>
<td>JST</td>
<td>JP</td>
<td>Japan Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Kwajalein</td>
<td>MHT</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Libya</td>
<td>EET</td>
<td>LY</td>
<td>Central European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MST</td>
<td></td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>MST7MDT</td>
<td>MDT</td>
<td></td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>PST</td>
<td>MX</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>Mexico/BajaNorte</td>
<td>PDT</td>
<td>MX</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MST</td>
<td>MX</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Mexico/BajaSur</td>
<td>MDT</td>
<td>MX</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Mexico/General</td>
<td>CST</td>
<td>MX</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Mexico/General</td>
<td>CDT</td>
<td>MX</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>NZ</td>
<td>NZST</td>
<td>NZ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>NZ</td>
<td>NZDT</td>
<td>NZ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>NZ-CHAT</td>
<td>CHAST</td>
<td>NZ</td>
<td>Chatham Standard Time</td>
<td>12:45</td>
</tr>
<tr>
<td>NZ-CHAT</td>
<td>CHADT</td>
<td>NZ</td>
<td>Chatham Daylight Time</td>
<td>13:45</td>
</tr>
<tr>
<td>Navajo</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Navajo</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>PRC</td>
<td>CST</td>
<td>CN</td>
<td>China Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>PST8PDT</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>PST8PDT</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>Poland</td>
<td>CET</td>
<td>PL</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Poland</td>
<td>CEST</td>
<td>PL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Portugal</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Portugal</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>ROC</td>
<td>CST</td>
<td>TW</td>
<td>Taipei Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>ROK</td>
<td>KST</td>
<td>KR</td>
<td>Korean Standard Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Singapore</td>
<td>SGT</td>
<td>SG</td>
<td>Singapore Standard Time</td>
<td>8:00</td>
</tr>
<tr>
<td>Turkey</td>
<td>EET</td>
<td>TR</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Turkey</td>
<td>EEST</td>
<td>TR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>UCT</td>
<td>UCT</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>US/Alaska</td>
<td>AKST</td>
<td>US</td>
<td>Alaska Standard Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>US/Alaska</td>
<td>AKDT</td>
<td>US</td>
<td>Alaska Daylight Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>US/Aleutian</td>
<td>HAST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>US/Aleutian</td>
<td>HADT</td>
<td>US</td>
<td>Hawaii-Aleutian Daylight Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>US/Arizona</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Central</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>US/Central</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/East-Indiana</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/East-Indiana</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>US/Eastern</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/Eastern</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>US/Hawaii</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>CST</td>
<td>US</td>
<td>Central Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>US/Indiana-Starke</td>
<td>CDT</td>
<td>US</td>
<td>Central Daylight Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>EST</td>
<td>US</td>
<td>Eastern Standard Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>US/Michigan</td>
<td>EDT</td>
<td>US</td>
<td>Eastern Daylight Time</td>
<td>-4:00</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>MST</td>
<td>US</td>
<td>Mountain Standard Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Mountain</td>
<td>MDT</td>
<td>US</td>
<td>Mountain Daylight Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>US/Pacific</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>PST</td>
<td>US</td>
<td>Pacific Standard Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>US/Pacific-New</td>
<td>PDT</td>
<td>US</td>
<td>Pacific Daylight Time</td>
<td>-7:00</td>
</tr>
<tr>
<td>US/Samoa</td>
<td>SST</td>
<td>AS</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>UTC</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Universal</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>W-SU</td>
<td>MSK</td>
<td>RU</td>
<td>Moscow Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Zulu</td>
<td>UTC</td>
<td></td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
</tbody>
</table>

**Area: Europe**

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe/Amsterdam</td>
<td>CEST</td>
<td>NL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CET</td>
<td>AD</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Andorra</td>
<td>CEST</td>
<td>AD</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Athens</td>
<td>EET</td>
<td>GR</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Athens</td>
<td>EEST</td>
<td>GR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Belfast</td>
<td>BST</td>
<td>GB</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CET</td>
<td>RS</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Belgrade</td>
<td>CEST</td>
<td>RS</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Berlin</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CET</td>
<td>SK</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Bratislava</td>
<td>CEST</td>
<td>SK</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CET</td>
<td>BE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Brussels</td>
<td>CEST</td>
<td>BE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EET</td>
<td>RO</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Bucharest</td>
<td>EEST</td>
<td>RO</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CET</td>
<td>HU</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Budapest</td>
<td>CEST</td>
<td>HU</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CET</td>
<td>DE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Busingen</td>
<td>CEST</td>
<td>DE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Chisinau</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CET</td>
<td>DK</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Copenhagen</td>
<td>CEST</td>
<td>DK</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>GMT</td>
<td>IE</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Dublin</td>
<td>IST</td>
<td>IE</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CET</td>
<td>GI</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Gibraltar</td>
<td>CEST</td>
<td>GI</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>GMT</td>
<td>GG</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Guernsey</td>
<td>BST</td>
<td>GG</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Helsinki</td>
<td>EET</td>
<td>FI</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Helsinki</td>
<td>EEST</td>
<td>FI</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>GMT</td>
<td>IM</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Isle_of_Man</td>
<td>BST</td>
<td>IM</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Istanbul</td>
<td>EET</td>
<td>TR</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Istanbul</td>
<td>EEST</td>
<td>TR</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>GMT</td>
<td>JE</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Jersey</td>
<td>BST</td>
<td>JE</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Kaliningrad</td>
<td>FET</td>
<td>RU</td>
<td>Eastern European Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Kiev</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WET</td>
<td>PT</td>
<td>Western European Standard Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/Lisbon</td>
<td>WEST</td>
<td>PT</td>
<td>Western European Summer Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CET</td>
<td>SI</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Ljubljana</td>
<td>CEST</td>
<td>SI</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/London</td>
<td>GMT</td>
<td>GB</td>
<td>Greenwich Mean Time</td>
<td>0:00</td>
</tr>
<tr>
<td>Europe/London</td>
<td>BST</td>
<td>GB</td>
<td></td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CET</td>
<td>LU</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Luxembourg</td>
<td>CEST</td>
<td>LU</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CET</td>
<td>ES</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Madrid</td>
<td>CEST</td>
<td>ES</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CET</td>
<td>MT</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Malta</td>
<td>CEST</td>
<td>MT</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EET</td>
<td>AX</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Mariehamn</td>
<td>EEST</td>
<td>AX</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Minsk</td>
<td>FET</td>
<td>BY</td>
<td>Eastern European Standard Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CET</td>
<td>MC</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Monaco</td>
<td>CEST</td>
<td>MC</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Moscow</td>
<td>MSK</td>
<td>RU</td>
<td>Moscow Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Europe/Nicosia</td>
<td>EET</td>
<td>CY</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>----------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Europe/Nicosia</td>
<td>EEST</td>
<td>CY</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Oslo</td>
<td>CET</td>
<td>NO</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Oslo</td>
<td>CEST</td>
<td>NO</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Paris</td>
<td>CET</td>
<td>FR</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Paris</td>
<td>CEST</td>
<td>FR</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Podgorica</td>
<td>CET</td>
<td>ME</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Podgorica</td>
<td>CEST</td>
<td>ME</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Prague</td>
<td>CET</td>
<td>CZ</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Prague</td>
<td>CEST</td>
<td>CZ</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Riga</td>
<td>EET</td>
<td>LV</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Riga</td>
<td>EEST</td>
<td>LV</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Rome</td>
<td>CET</td>
<td>IT</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Rome</td>
<td>CEST</td>
<td>IT</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Samara</td>
<td>SAMT</td>
<td>RU</td>
<td>Moscow Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Europe/San_Marino</td>
<td>CET</td>
<td>SM</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/San_Marino</td>
<td>CEST</td>
<td>SM</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CET</td>
<td>BA</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Sarajevo</td>
<td>CEST</td>
<td>BA</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Simferopol</td>
<td>MSK</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CET</td>
<td>MK</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Skopje</td>
<td>CEST</td>
<td>MK</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EET</td>
<td>BG</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Sofia</td>
<td>EEST</td>
<td>BG</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CET</td>
<td>SE</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Stockholm</td>
<td>CEST</td>
<td>SE</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EET</td>
<td>EE</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tallinn</td>
<td>EEST</td>
<td>EE</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Tirane</td>
<td>CET</td>
<td>AL</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>--------</td>
<td>---------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Europe/Tirane</td>
<td>CEST</td>
<td>AL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EET</td>
<td>MD</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Tiraspol</td>
<td>EEST</td>
<td>MD</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Uzhgorod</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CET</td>
<td>LI</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Vaduz</td>
<td>CEST</td>
<td>LI</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CET</td>
<td>VA</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Vatican</td>
<td>CEST</td>
<td>VA</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CET</td>
<td>AT</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Vienna</td>
<td>CEST</td>
<td>AT</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EET</td>
<td>LT</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Vilnius</td>
<td>EEST</td>
<td>LT</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Volgograd</td>
<td>VOLT</td>
<td>RU</td>
<td>Volgograd Standard Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CET</td>
<td>PL</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Warsaw</td>
<td>CEST</td>
<td>PL</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CET</td>
<td>HR</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Zagreb</td>
<td>CEST</td>
<td>HR</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EET</td>
<td>UA</td>
<td>Eastern European Standard Time</td>
<td>2:00</td>
</tr>
<tr>
<td>Europe/Zaporozhye</td>
<td>EEST</td>
<td>UA</td>
<td>Eastern European Summer Time</td>
<td>3:00</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CET</td>
<td>CH</td>
<td>Central European Standard Time</td>
<td>1:00</td>
</tr>
<tr>
<td>Europe/Zurich</td>
<td>CEST</td>
<td>CH</td>
<td>Central European Summer Time</td>
<td>2:00</td>
</tr>
</tbody>
</table>

**Area: Pacific**

<table>
<thead>
<tr>
<th>Time Zone Information</th>
<th>Time Zone Name</th>
<th>Region</th>
<th>Time Zone Description</th>
<th>Time Zone Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pacific/Apiap</td>
<td>WST</td>
<td>WS</td>
<td>Samoa Standard Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Pacific/Apia</td>
<td>WSDT</td>
<td>WS</td>
<td>Samoa Daylight Time</td>
<td>14:00</td>
</tr>
<tr>
<td>Pacific/Auckland</td>
<td>NZST</td>
<td>NZ</td>
<td>New Zealand Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Auckland</td>
<td>NZDT</td>
<td>NZ</td>
<td>New Zealand Daylight Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Chatham</td>
<td>CHAST</td>
<td>NZ</td>
<td>Chatham Standard Time</td>
<td>12:45</td>
</tr>
<tr>
<td>Pacific/Chatham</td>
<td>CHADT</td>
<td>NZ</td>
<td>Chatham Daylight Time</td>
<td>13:45</td>
</tr>
<tr>
<td>Pacific/Chuuk</td>
<td>CHUT</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Easter</td>
<td>EAST</td>
<td>CL</td>
<td>Easter Island Standard Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Pacific/Easter</td>
<td>EASST</td>
<td>CL</td>
<td>Easter Island Summer Time</td>
<td>-5:00</td>
</tr>
<tr>
<td>Pacific/Efate</td>
<td>VUT</td>
<td>VU</td>
<td>Vanuatu Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Enderbury</td>
<td>PHOT</td>
<td>KI</td>
<td>Phoenix Islands Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Fakaofu</td>
<td>TKT</td>
<td>TK</td>
<td>Tokelau Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Fiji</td>
<td>FJT</td>
<td>FJ</td>
<td>Fiji Standard Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Funafuti</td>
<td>TVT</td>
<td>TV</td>
<td>Tuvalu Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Galapagos</td>
<td>GALT</td>
<td>EC</td>
<td>Galapagos Time</td>
<td>-6:00</td>
</tr>
<tr>
<td>Pacific/Gambier</td>
<td>GAMT</td>
<td>PF</td>
<td>Gambier Time</td>
<td>-9:00</td>
</tr>
<tr>
<td>Pacific/Guadalcanal</td>
<td>SBT</td>
<td>SB</td>
<td>Solomon Islands Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Guam</td>
<td>CHST</td>
<td>GU</td>
<td>Chamorro Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Honolulu</td>
<td>HST</td>
<td>US</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Johnston</td>
<td>HST</td>
<td>UM</td>
<td>Hawaii-Aleutian Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Kiritimati</td>
<td>LINT</td>
<td>KI</td>
<td>Line Islands Time</td>
<td>14:00</td>
</tr>
<tr>
<td>Pacific/Kosrae</td>
<td>KOST</td>
<td>FM</td>
<td>Kosrae Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Kwajalein</td>
<td>MHT</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Majuro</td>
<td>MHT</td>
<td>MH</td>
<td>Marshall Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Marquesas</td>
<td>MART</td>
<td>PF</td>
<td>Marquesas Time</td>
<td>-6:27</td>
</tr>
<tr>
<td>Pacific/Midway</td>
<td>SST</td>
<td>UM</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Nauru</td>
<td>NRT</td>
<td>NR</td>
<td>Nauru Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Niue</td>
<td>NUT</td>
<td>NU</td>
<td>Niue Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Time Zone Information</td>
<td>Time Zone Name</td>
<td>Region</td>
<td>Time Zone Description</td>
<td>Time Zone Offset</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Pacific/Norfolk</td>
<td>NFT</td>
<td>NF</td>
<td>Norfolk Island Time</td>
<td>11:30</td>
</tr>
<tr>
<td>Pacific/Noumea</td>
<td>NCT</td>
<td>NC</td>
<td>New Caledonia Standard Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Pago_Pago</td>
<td>SST</td>
<td>AS</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Palau</td>
<td>PWT</td>
<td>PW</td>
<td>Palau Time</td>
<td>9:00</td>
</tr>
<tr>
<td>Pacific/Pitcairn</td>
<td>PST</td>
<td>PN</td>
<td>Pitcairn Time</td>
<td>-8:00</td>
</tr>
<tr>
<td>Pacific/Pohnpei</td>
<td>PONT</td>
<td>FM</td>
<td>Ponape Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Ponape</td>
<td>PONT</td>
<td>FM</td>
<td>Ponape Time</td>
<td>11:00</td>
</tr>
<tr>
<td>Pacific/Port_Moresby</td>
<td>PGT</td>
<td>PG</td>
<td>Papua New Guinea Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Rarotonga</td>
<td>CKT</td>
<td>CK</td>
<td>Cook Islands Standard Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Saipan</td>
<td>CHST</td>
<td>MP</td>
<td>Chamorro Standard Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Samoa</td>
<td>SST</td>
<td>AS</td>
<td>Samoa Standard Time</td>
<td>-11:00</td>
</tr>
<tr>
<td>Pacific/Tahiti</td>
<td>TAHT</td>
<td>PF</td>
<td>Tahiti Time</td>
<td>-10:00</td>
</tr>
<tr>
<td>Pacific/Tarawa</td>
<td>GILT</td>
<td>KI</td>
<td>Gilbert Islands Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Tongatapu</td>
<td>TOT</td>
<td>TO</td>
<td>Tonga Standard Time</td>
<td>13:00</td>
</tr>
<tr>
<td>Pacific/Truk</td>
<td>CHUT</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
<tr>
<td>Pacific/Wake</td>
<td>WAKT</td>
<td>UM</td>
<td>Wake Island Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Wallis</td>
<td>WFT</td>
<td>WF</td>
<td>Wallis and Futuna Time</td>
<td>12:00</td>
</tr>
<tr>
<td>Pacific/Yap</td>
<td>CHUT</td>
<td>FM</td>
<td>Chuuk Time</td>
<td>10:00</td>
</tr>
</tbody>
</table>
Recommended Reading

Here is the recommended reading list for this title:

- *Base SAS Procedures Guide*
- *SAS Companion for your operating environment*
- *SAS/CONNECT User's Guide*
- *SAS Data Set Options: Reference*
- *SAS Formats and Informats: Reference*
- *SAS Functions and CALL Routines: Reference*
- *SAS/GRAPH: Reference*
- *SAS Language Reference: Concepts*
- *SAS System Options: Reference*
- *SAS Statements: Reference*
- *SAS Encoding - Understanding the Details*

For a complete list of SAS publications, go to [sas.com/store/books](http://sas.com/store/books). If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: [sas.com/store/books](http://sas.com/store/books)
Glossary

ActiveX
a technology developed by Microsoft that is used to add interactivity to web pages.

annotation
a label, marker, or note that is not obtained from the data but is placed on a graph independently. Such annotations might or might not be linked to data values in the plot.

axis
a line that represents the midpoints (for a discrete axis) or the scale (for a continuous or interval axis) for graphing variable or data values. An axis typically consists of an axis line with tick marks, tick values (or midpoint values), and a label.

block
See statement block.

cell
See graph cell.

class variable
See classification variable.

classification panel
a multi-cell graph in which the cell data is driven by the values of one or more classification variables. The number of the cells is determined by the unique values of the classification variables. Each cell of the panel has the same types of plots.

classification variable (class variable)
a variable whose values are used to classify the observations in a data set into different groups that are meaningful for analysis. A classification variable can have either character or numeric values. Classification variables include group, subgroup, category, and BY variables.

computed plot
a plot in which input data is internally summarized or otherwise transformed to create new data that is actually rendered by the plot. Examples of computed plot statements are BARCHART, BOXPLOT, HISTOGRAM, ELLIPSE, and REGRESSIONPLOT.
define block
in the TEMPLATE procedure, a define block (beginning with a DEFINE statement and ending with an END statement) creates various types of templates, including STATGRAPH, STYLE, and TABLE.

dependent plot
a plot that cannot be rendered by itself. Dependent plots must be overlaid with a stand-alone plot. Dependent plots do not provide data ranges to establish axes. REFERENCELINE, DROPLINE, and LINEPARM statements produce dependent plots.

destination
See ODS destination.

device-based graphic
a graph created with SAS/GRAPH software for which a user-specified or default device (DEVICE= option) controls certain aspects of the graphical output.

dots per inch (DPI)
a measure of the graph resolution by its dot density.

DPI
See dots per inch.

dynamic variable
a variable defined in a template with the DYNAMIC statement that can be initialized at template run time.

fill
to apply a color within a bounded area. Many plots, such as bar charts and band plots, have bounded areas that can be filled or unfilled. When filled, a color is applied. When unfilled, the areas are transparent.

footnote area
the region below the graph area where text produced by ENTRYFOOTNOTE statements appears.

graph cell (cell)
a distinct rectangular subregion of a graph that can contain plots, text, or legends.

Graph Gallery
in ODS Graphics Designer, a gallery of predefined, commonly used graphs. Users can add their own custom graphs to the gallery.

graph panel
a graph with multiple cells.

Graph Template Language (GTL)
an extension to the Output Delivery System (ODS) that enables users to create sophisticated analytical graphs.

graphics template
See ODS template.
group variable
a variable in the input data set that is used to categorize chart variable values into groups. A group variable enables the data for each distinct group value to be rendered in a visually different manner. For example, a grouped scatter plot displays a distinct marker and color for each group value.

GTL
See Graph Template Language.

layout
a generic term for a rectangular container that lays out the positions and sizes of its child components.

layout block
a block beginning with a LAYOUT statement and ending with an ENDLAYOUT statement.

layout grid
a multi-cell layout arranged as a grid of cells in rows and columns.

layout row (row)
a set of layout cells that are side-by-side and share the same alignment.

layout type
a keyword indicating the functionality of the layout. For example OVERLAY, LATTICE, and DATAPANEL are layout types.

legend entry
a combination of a graphical element such as a marker or line along with text describing the value or use of the graphical element. A discrete legend can have several legend entries.

loess plot
a curved line showing a loess fit for a set of points.

marker
a symbol such as a diamond, a circle, or a triangle that is used to indicate the location of, or annotate, a data point in a plot or graph.

multi-cell layout
a layout that supports a rectangular grid of cells, each of which can contain a graphical element, such as a plot, a legend, a nested layout, and so on.

nested layout
a layout block that appears within the scope of another layout block.

ODS
See Output Delivery System.

ODS destination (destination)
a designation that the Output Delivery System uses to generate a specific type of output. Types of ODS destinations include but are not limited to HTML, XML, listing, PostScript, RTF, and SAS data sets.
ODS Graphics
an extension to ODS that is used to create analytical graphs using the Graph Template Language.

ODS style (style)
a combination of colors, fonts, lines, marker symbols, and so on that provide a specific appearance for SAS output. A style is defined in ODS by a style template.

ODS table
a table that is created with an IMSTAT procedure statement with the SAVE= option. The contents of the table resemble ODS output and are commonly used with the STORE statement. This enables the ODS output from one statement to be used as input for another statement.

ODS template (graphics template)
a description of how output should appear when it is formatted. ODS templates are stored as compiled entries in a template store, also known as an item store. Common template types include STATGRAPH, STYLE, CROSSTABS, TAGSET, and TABLE.

Output Delivery System (ODS)
a component of SAS software that can produce output in a variety of formats such as markup languages (HTML, XML), PDF, listing, RTF, PostScript, and SAS data sets.

overlay
a plot that can be superimposed on another plot when specified within an overlay-type layout. A common overlay combination is a fit line on a scatter plot.

overlay layout
a type of layout that supports the superimposition of graphical components, such as plots, legends, and nested layouts.

plot
a visual representation of data such as a scatter plot, needle plot, or contour plot.

plot area
the space, bounded by the axes, where a visual representation of data, such as a scatter plot, a series line, or a histogram, is drawn.

plot type
a plot family such as bar chart (which would include horizontal, vertical, and grouped bar charts), or a classification scheme for plots based on some useful criteria, such as whether the plots are computed or parameterized.

regression plot
a straight or curved line showing a linear or higher order regression fit for a set of points.

required argument
a variable or constant that must be specified in order to evaluate an expression or render a plot, legend, text, or a layout. For example, a scatter plot has two required arguments: X=column and Y=column.

results table
an ODS table object that is used with the STORE, REPLAY, and FREE statements. It is similar to creating an output data set from an ODS table because ODS tables are
used. It is different because the table is not written to disk as a data set, but is
available in the SAS session until the IMSTAT procedure ends.

row
   See layout row.

scatter plot matrix
   a grid of scatter plots showing pairwise combinations of multiple numeric variables.

SGD file
   an ODS Graphics Designer file. Users can open this file in the designer and change
   the graph. Users can also render the graph to an ODS destination by using the
   SGDESIGN procedure.

SGE file
   a file created in the ODS Graphics environment that contains an editable graph. Such
   files have a .sge file extension and can be edited only with the ODS Graphics Editor.
   You can edit SGE files from the SAS Results window or by opening the SGE file
   from within the ODS Graphics Editor.

shared variable
   a feature of ODS Graphics Designer that enables users to reuse graphs and specify
   different variables from the same or from a different data set.

single-cell layout
   a layout type that supports only one cell. The OVERLAY, OVERLAY3D, and
   OVERLAYEQUATED layouts are examples of single-cell layouts.

stand-alone plot
   a plot that has its own data range and can therefore appear by itself in a layout.

statement block (block)
   a group of statements that has both a logical beginning and ending statement. For
   example, a LAYOUT statement along with its ENDLAYOUT statement and all
   contained statements are a block. Some blocks can be nested within other blocks.

style
   See ODS style.

style attribute
   a visual property, such as color, font properties, and line characteristics, that is
   defined in ODS with a reserved name and value. Style attributes are collectively
   referenced by a style element within a style template.

style element
   a named collection of style attributes that affects specific parts of ODS output. For
   example, a style element might specify the color and font properties of title text or
   other text in in a table or graph. See also style attribute.

style template
   a type of ODS template that defines the visual aspects (colors, fonts, lines, markers,
   and so on) of SAS output for a specific style. A style template defines style elements,
   and each style element consists of style attributes.
**template definition (template source)**
the TEMPLATE procedure source program that creates a template. A template definition can be generated from a compiled template. Also called the template source.

**template source**
See template definition.

**template store**
an item store that contains definitions that were created by the TEMPLATE procedure. Definitions that SAS provides are in the item store Sashelp.Tmplmst. You can store definitions that you create in any template store to which you have Write access.

**template-based graphic**
graphical output produced by a compiled ODS template of the type STATGRAPH. That is, a graph that is produced within the ODS graphics environment rather than in the traditional device-based environment.

**transparency**
the degree to which a graphic element (such as a marker or filled area) is opaque or transparent. Transparency is indicated with a number from 0 (completely opaque) to 1 (completely transparent).

**Unicode**
a 16-bit encoding that is the industry standard for supporting the interchange, processing, and display of characters and symbols from most of the world's writing systems.
Index

Special Characters
SBIDlw. format 105
SCPTDlw. format 107
SCPTWDw. format 108
SEBCDIClw. format 109
SKANJlw. format 121
SKANJlw. informat 440
SKANJXlw. format 122
SKANJXlw. informat 441
SLOGVSWw. format 123
SLOGVSWw. informat 536
SREVERSJw. informat 537
SREVERSJw. informat 537 compared to SREVERSw. informat 537
SUCS2BEw. format 268
SUCS2BEw. informat 539
SUCS2Bw. format 266
SUCS2Bw. informat 538
SUCS2LEw. format 270
SUCS2LEw. informat 541
SUCS2Lw. format 269
SUCS2Lw. informat 540
SUCS2XEw informat 537
SUCS2XEw. format 273
SUCS2Xw. format 271
SUCS2Xw. informat 542
SUCS4BEw. format 275
SUCS4Bw informat 544
SUCS4Bw. format 274
SUCS4LEw. format 278
SUCS4Lw informat 545
SUCS4Lw. format 276
SUCS4XEw. format 280
SUCS4XEw. informat 548
SUCS4Xw. format 279
SUCS4Xw. informat 546
SUESCEw. format 283
SUESCEw. informat 550
SUESCw. format 281
SUESCw. informat 549
SUNCRew. format 285
SUNCRew. informat 552
SUNCRew. format 284
SUNCRew. informat 551
SUPARENEw. format 287
SUPARENEw. informat 554
SUPARENEw. format 555
SUPARENEw. format 286
SUPARENEw. informat 553
$UTF8XEw.format 289
$UTF8Xw. format 288
$UTF8Xw. informat 557
$VSLOGRSw. format 291
$VSLOGRSw. informat 559
$VSLOGw. format 290
$VSLOGw. informat 558
%KCMPRES 566
%KINDEX Macro Function 566
%KLEFT macro function 567
%KLENGTH macro function 567
%KLOWCASE autocall macro 51
%KSCAN macro function 568
%KSUBSTR macro function 570
%KTRIM autocall macro 52
%KVERIFY autocall macro 52
%QKCMPPRES 566
%QKTRIM autocall macro 52

Numbers
8859 ISO family 13

A
alignment character expressions 351, 361
ANORM420 Function 330
ANSI (American National Standards Institute) 15
Arabic characters reversing 105
storing logical-ordered text on visual server 123, 124
arguments converting to lowercase 353
converting to uppercase 372
extracting a substring from 366
extracting a substring from, based on byte position 367
length of 352
transcoding for specified argument 420
ASCII 13
transferring data between EBCDIC and 32
ATTRIB statement 640
TRANSCODE= option 641
Australia
monetary format 177, 214
Austria
monetary format 224
Austrian schillings
converting to euros 749

B
B8601DXw. format 99
B8601LXw. format 101
B8601TXw. format 102
BASECHAR function 334
BASETYPE= option
PROC DBCSTAB statement 650
Belgium
monetary format 187, 224
BESTDOTX format 104
binary collation 17
blanks
removing leading DBCS blanks 351
trimming trailing 52
trimming trailing DBCS blanks 361
BOMFILE system option 579
BOTH option
CLEAR statement (TRANTAB) 674
LIST statement (TRANTAB) 675
SAVE statement (TRANTAB) 678
Brazil
monetary format 179, 216
Bulgaria
monetary format 178, 215
Byte Order Mark (BOM) prefix
on Unicode external files 579

C
Canada
monetary format 180, 217
case
changing uppercase characters to lowercase 51
CATALOG= option
PROC DBCSTAB statement 650
CEDA 30
character data
converting to EBCDIC 109
reading from right to left 536, 537
character expressions
comparing 342
compressing 343
concatenating 363
deleting character value contents 373
deleting character value contents, based on byte unit 375
inserting character value contents 373
inserting character value contents, based on byte unit 375
left-aligning 351
number of double-byte characters in position of first unique character 378
removing trailing blanks and SO/SI 370
replacing character value contents 373
replacing character value contents, based on byte unit 375
replacing specific characters 368
reversing 360
right-aligning 361
searching for specific characters 350
searching for string of characters 347
selecting a specified word from 362
translating 368
trimming 370
updating 373
updating, based on byte unit 375
verifying 378
character sets 12
definition 10
displaying DBCS 38
specifying, for META declaration in output 616
translation tables and 670
character strings 570
split 40
character variables
transcoding enabled for specified variable 419
character-set encoding
for SAS session 587
characters locating 566
CHARSET= option 616
China
monetary format 182, 219
CIMPORT procedure
transcoding between operating environments 30
CLEAR statement 674
TRANTAB procedure 674
code page 9
collating sequence 17
alternate sequences 18
binary collation 17
encoding value 19
language-specific 602
linguistic collation 20
overview 17
results of different sequences 17
translation tables 19
commas, removing 560
compatibility
  SAS string functions 307
compatible encodings 32
concatenation
  character expressions 363
CONTENTS statement 662
conversion tables
  creating 651
  for DBCS 649
  Japanese 653
CORRECTENCODING option 624
CPORT procedure
  transcoding between operating environments 30
CPTDw.w informat 435
CPTDw informat 436
Croatia
  monetary format 190, 227
cross-environment data access (CEDA) 30
currency 6
  converting one European currency to another 816
  yen 297
CVPBYTES option 625
Czech Republic
  monetary format 183, 220

D
data conversion
  between DBCS encodings 40
data set options
  for transcoding 30
data sets
  encoding 21
  encoding support, by release 25
  mixed encodings 59
  suppressing transcoding 59
  transcode attributes of variables 417
  with a particular encoding 59
DATA= option
  PROC DBCCSTAB statement 650
date format descriptors 380
date values
  as a date 130
  converting to specified locale 380
  date and day of week 139
day of week 140
Hebrew 119
international 811
international month and year 814
Japanese 127, 446
Jewish calendar 120
name and day of month 133
name of month 137
Taiwanese 126, 444
year 149
year and name of month 141
year and quarter 145
year and week 150
dates 5
  date values as 130
DATESTYLE= system option 580
default values 704
datetime values
  as a datetime 151
  B8601DT format, ISO 8601 basic notation, no time zone 99, 101
  B8601TX format, ISO 8601 basic notation, time zone offset 102
  converting to specified locale 383, 386
day of week 164
day of week and datetime 163
e8601DT format, ISO 8601 basic notation, no time zone 110, 111
e8601TX format, ISO 8601 basic notation, time zone offset 113
international 812
  name and day of month 155
  name of month 159
  name of month, day of month and year 153
time of day 161
  with a.m. or p.m. 152
  year 173
  year and name of month 166
  year and name of week 174
  year and quarter 169
datetime-format descriptors 383
DBCS
  See double-byte character sets (DBCS)
DBCS data
  adding shift-code data to 121, 441
  removing shift-code data from 122, 440
DBCS encoding 12, 37
  See also double-byte character sets (DBCS)
  character data truncation 34
data conversion between encodings 40
East Asian languages 33
encoding values for transcoding data 713
full-screen 589
full-screen input method modules (IMMs) 590, 591
removing leading blanks 351
requirements for displaying character sets 38
SAS on mainframe and specifying 38
split character strings 40
system option values for 711
system options in SAS session for trimming blanks and SO/SI 370
trimming trailing blanks 361
when you can use 39
DBCS system option 581
DBCSLANG system option 582
DBCSLANG= option
PROC DBCSTAB statement 650
DBCSTAB procedure 649
creating conversion tables 651
examples 651
Japanese conversion tables 653
PROC DBCSTAB statement 650
syntax 649
DBCSTAB statement 649
DBCTYPE system option 583
decimal points, removing 560
Denmark
monetary format 184, 221
DESC= option
PROC DBCSTAB statement 650
DFLANG= system option 585
default values 704
double-byte character sets (DBCS) 12, 581
See also DBCS encoding
collection tables for 649
encoding method 583
language for 582
recognizing 581
double-byte characters
number in a character expression 344

Egypt
monetary format 186, 223
ENCODCOMPAT function 336
encoding 9
behavior in SAS sessions 25
character sets for 12
compatibility for transcoding 32
converting one type of data to another 345
data set support by release 25
data sets 21
default SAS session encoding 22
definition 11
for East Asian languages 13
input processing 25, 60
mixed 59
output processing 25
overriding 57
reading and writing external files 26
SAS sessions 21
setting for SAS sessions 23
standards organizations for 15
versus transcoding 12
encoding methods 10, 13
for DCBS 583
ENCODING option 630
ENCODING system option 587
encoding values 19, 713
DBCS 713
default SAS session values 22
default, based on LOCALE= system option 24
for transcoding data 713
SBCS 713
Unicode 713
UNIX 723
Windows 724
z/OS 725
ENCODING= data set option 57
ENCODING= system option
default settings 704
Posix values 704
ENCODISVALID Function 337
escape codes 39
Estonia
monetary format 185, 222
EUR language elements 732
EURDFDDw. format 732
EURDFDEw. format 734
EURDFDEw. informat 811
EURDFDNw. format 735
EURDFDTw. format 736
EURDFDTw. informat 812
EURDFDWNw. format 738
EURDFMnw. format 741
EURDFMY format
mmmyy  742
EURDFMYw. format  742
EURDFMYw. format  814
EURDFWDXw. format  744
EURDFWXKw. format  746
EURFRATSw.d format  749
EURFRBEFw.d format  750
EURFRCFFw.d format  752
EURFRDEMw.d format  753
EURFRDXXw. format  754
EURFRESPw.d format  756
EURFRFIMw.d format  757
EURFRFRFw.d format  758
EURFRGBPw.d format  759
EURFRGRDw.d format  761
EURFRHUFw.d format  762
EURFRIEPw.d format  763
EURFRITLw.d format  765
EURFRLUFW.d format  766
EURFRNLGw.d format  767
EURFRNOKw.d format  769
EURFRPLZw.d format  770
EURFRPTw.d format  771
EURFRROLw.d format  773
EURFRRUw.d format  774
EURFRSEKw.d format  775
EURFRSITw.d format  777
EURFRTLXw.d format  778
euro conversion
   Austrian schillings to euros  749
EUROCURRE function  816
European currency conversion
   converting one currency to another  816
   euros
   formats for  114, 117
EUROW.d format  114
EUROW.d informat  437
EUROWXw. format  117
EUROWXw. informat  439
EURTOATSw.d format  779
EURTOBEFw.d format  780
EURTOCHFw.d format  782
EURTOFKw.d format  783
EURTODEMw. format  784
EURTODKKw.d format  786
EURTOESPw.d format  787
EURTOFMw.d format  788
EURTOFRFw.d format  790
EURTOGBPw.d format  791
EURTOWRDw.d format  792
EURTOHUFw.d format  794
EURTOIEPw.d format  795
EURTOITLw.d format  796
EURTOLUW.d format  798
EURTONLGw.d format  799
EURTONOKw.d format  800
EURTOPLZw.d format  802
EURTOPTw.d format  803
EURTOROLw.d format  804
EURTORowo.d format  806
EURTOSEKw.d format  807
EURTOSITw.d format  808
EURTOTRLw.d format  810
external files
   BOM prefix on Unicode files  579
   encoding and  26

F
Faroe Island
   monetary format  184, 221
filerefs
   limiting to eight characters  570
Finland
   monetary format  187, 224
FORCE option
   PROC DBCSTAB statement  650
formats
   associating with variables  640
   international date and datetime formats  63
   language for international dates  585
   NLS  63
   supporting DBCS on SO/SI systems  39
France
   monetary format  187, 224
   FSDBTYPE system option  589
   FSIMM system option  590
   FSIMMOPT system option  591
   full-screen DBCS encoding  589
   input method modules (IMMs) for  590, 591
functions
   by category  326
   K functions  307
   SAS string functions  307

G
Germany
   monetary format  187, 224
GETLOCENV function  338
GETPXLANGUAGE function  339
GETPXLOCALE function  340
GETPXREGION function  341
Greece
   monetary format  187, 224
Greenland
   monetary format  184, 221
ISO (International Organization for Standardization) 15
ISO 8601 date and time formats
B8601DX format, basic datetime notation, time zone offset 99
B8601LX format, basic datetime notation, time zone offset 101
B8601TX format, basic time notation, time zone offset 102
E8601DX format, basic datetime notation, time zone offset 110
E8601LX format, basic datetime notation, time zone offset 111
E8601TX format, basic time notation, time zone offset 113
ISO encodings 13
8859 family 13
Windows family 13
Israel monetary format 193, 230
Italy monetary format 187, 224
item stores naming 607
Japan monetary format 195, 232
Japanese conversion tables 653
Japanese dates 127, 446
Jewish calendar dates 120
K functions 307
KCOMPARE function 342
KCOMPRESS function 343
KCOUNT function 344
KCVT function 345
KINDEX function 347
KINDEXB function 348
KINDEXC function 349
KINDEXCB function 350
KLEFT function 351
KLENGTH function 352
KLOWCASE function 353
KPROPCASE function 354
KPROPCHAR function 357
KPROPIDATA function 358
KREVERSE function 360
KRIGHT function 361
KSCAN function 362
KSTRCAT function 363
KSTRIP function 364
KSUBSTR function 366
H
HDATElw. format 119
HEBDATElw. format 120
Hebrew characters 107, 108
reversing 105
storing logical-ordered text on visual server 123, 124
Hebrew date values 119
hexadecimal representation translation tables in 675
Hong Kong monetary format 189, 226
Hungary monetary format 191, 228
I
IBw.d informat 526, 527, 529, 530
illegal data 601
incompatible encodings 32
India monetary format 194, 231
Indonesia monetary format 192, 229
INENCODING= and OUTENCODING= option 634
informs ambiguous data 580
associating with variables 640
language for international dates 585
supporting DBCS on SO/SI systems 39
input method modules (IMMs) 590
options for 591
input processing 25
overriding encoding for 60
integer binary values, reading 526, 527, 529, 530
international date and datetime formats 63
international date formats and informats specifying language for 585
international date values, writing
day-of-week and date 746
day-of-week name 738
day-of-week number 735
dd.mm.yy 732
ddmmmyy 734
month name 741, 744
international datet ime values, writing
ddmmmyy:hh:mm:ss:ss 736
International Organization for Standardization (ISO) 15
internationalization 4
INVERSE statement 675
Ireland monetary format 187, 224
KSUBSTRB function 367
KTRANSLATE function 368
KTRIM function 370
KTRUNCATE function 371
KUPCASE function 372
KUPCASE macro function 572
KUPDATE function 373
KUPDATEB function 375
KUPDATES function 376
KVERIFY function 378
KVERIFYB function 379

labels, associating with variables 640
language 5
language codes
current two-letter code 339
language switching 8
changing text language of ODS output 599
languages
for international date informatns and formats 585
Latin1 code page 9
Latvia
monetary format 198, 235
left-aligning character expressions 351
length
associating with variables 640
of arguments 352
Liechtenstein
monetary format 181, 218
line-feed characters 32
linguistic collation 20
linguistic sort keys 392
LIST statement 675
TRANTAB procedure 675
Lithuania
monetary format 197, 234
LOAD statement 656, 676
TRANTAB procedure 676
locale 5
best numerical notation based on 129
converting date values to specified locale 380
converting datetime values to specified locale 383
converting time or datetime values to specified locale 386
language switching 8
of SAS sessions 591
POSIX value for 340
specifying 6
specifying at SAS invocation 6
specifying during SAS session 7
locale data 663
LOCALE system option 591
LOCALE= system option
default encoding values based on 24
values for 697
LOCALEDATA procedure 655, 656
LOCALEDATA Procedure 655
LOCALEDATA system option 593
localization 4
logical-ordered text
storing on visual server 123, 124
LOGLANGCHG system option 594
LOGLANGENG system option 595
LOGVSRw. informat 443
LOGVSW. informat 442
long macro variables
storing values in segments 571
lowercase characters
changing uppercase characters to 51
converting arguments to 353
LSWLANG system option 596
Luxembourg
monetary format 187, 224

Macau
monetary format 199, 236
macro variables
storing long values in segments 571
mainframes
DBCS and 39
Malaysia
monetary format 201, 238
Malta
monetary format 187, 224
MAPEBCDIC2ASCII system option 597
MBCS encoding 12
META declaration 616
Mexico
monetary format 200, 237
MINGUOw. format 126
MINGUOw. informat 444
MODIFY statement 656
monetary formats
Australia 177, 214
Austria 224
Belgium 187, 224
Brazil 179, 216
Bulgaria 178, 215
Canada 180, 217
China 182, 219
Croatia 190, 227
Czech Republic 183, 220
Denmark 184, 221
Egypt 186, 223

Index 865
Estonia 185, 222
Faroe Island 184, 221
Finland 187, 224
France 187, 224
Germany 187, 224
Greece 187, 224
Greenland 184, 221
Hong Kong 189, 226
Hungary 191, 228
India 194, 231
Indonesia 192, 229
Ireland 187, 224
Israel 193, 230
Italy 187, 224
Japan 195, 232
Latvia 198, 235
Liechtenstein 181, 218
Lithuania 197, 234
Luxembourg 187, 224
Macau 199, 236
Malaysia 201, 238
Malta 187, 224
Mexico 200, 237
Netherlands 187, 224
New Zealand 203, 240
Norway 202, 239
Poland 204, 241
Portugal 187, 224
Puerto Rico 211, 248
Russia 205, 242
Singapore 207, 244
Slovenia 187, 224
South Africa 212, 249
South Korea 196, 233
Spain 187, 224
Sweden 206, 243
Switzerland 181, 218
Taiwan 210, 247
Thailand 208, 245
Turkey 209, 246
United Arab Emirates 176, 213
United Kingdom 188, 225
United States 211, 248

N
National Language Support
See NLS (National Language Support)
National Language Support (NLS) formats 63
NENGOw. format 127
NENGOw. informat 446
Netherlands monetary format 187, 224
New Zealand monetary format 203, 240
NLBESTw. format 129
NLDATE function 380
NLDATEL format 132
NLDATEM format 132
NLDATEMDL Format 134
NLDATEMDM format 135, 157
NLDATEMS format 136
NLDATEMDw. format 133
NLDATENw. format 137
NLDATEMYL format 142
NLDATES format 138
NLDATEW informat 448
NLDATEw. format 130
NLDATEw. informat 447
NLDATENw. format 140
NLDATEWw. format 139
NLDATEMYMM format 143
NLDATEMYMS format 144
NLDATEMYw. format 141
NLDATEMYQL format 146
NLDATEMYQM format 147
NLDATEMYQS format 148
NLDATEMYQw. format 145
NLDATEMYRw. format 149
NLDATEMYWw. format 150
NLDATMAP informat 450
NLDATMAPw. format 152
NLDATMDTw. format 153
NLDATML format 154
NLDATMM format 155
NLDATMMMDL format 156
NLDATMMDS format 158
NLDATMMMDw. format 155
NLDATMMNw. format 159
NLDATMS format 160
NLDATMTMw. format 161
NLDATMTZw. format
datetime values 162
NLDATMW informat 451
NLDATMw. format 151
NLDATMw. informat 449
NLDATMWNw. format 164
NLDATMWW. format 163
NLDATMZWw. format
datetime values 165
NLDATMYML format 167
NLDATMYMM format 168
NLDATMYMS format 168
NLDATMYMw. format 166
NLDATMYQL format 170
NLDATMYQM format 171
NLDATMYQS format 172
NLDATMYQw. format 169
NLDATMYRw. format 173
NLDATMYWw. format 174
NLDATMZw. format
datetime values 175

NLDECSEPARATOR system option 598

NLMNIAEDw.d format 176
NLMNIAEDw.d informat 452
NLMNIAUDw.d format 177
NLMNIAUDw.d informat 453
NLMNIBGNw.d format 178
NLMNIBGNw.d informat 454
NLMNIBRLw.d format 179
NLMNIBRLw.d informat 455
NLMNICADw.d format 180
NLMNICADw.d informat 456
NLMNICFw.d format 181
NLMNICFw.d informat 457
NLMNICNYw.d format 182
NLMNICNYw.d informat 458
NLMNICZKw.d format 183
NLMNICZKw.d informat 459
NLMNIDKKw.d format 184
NLMNIDKKw.d informat 460
NLMNIEEKw.d format 185
NLMNIEEKw.d informat 461
NLMNIEGPw.d format 186
NLMNIEGPw.d informat 462
NLMNIEURw.d format 187
NLMNIEURw.d informat 463
NLMNIGBPw.d format 188
NLMNIGBPw.d informat 464
NLMNIIHKDw.d format 189
NLMNIIHKDw.d informat 465
NLMNIIHRKw.d format 190
NLMNIIHRKw.d informat 466
NLMNIIHUw.d format 191
NLMNIIHUw.d informat 467
NLMNIIIDRw.d format 192
NLMNIIIDRw.d informat 468
NLMNIILSw.d format 193
NLMNIILSw.d informat 469
NLMNIINRw.d format 194
NLMNIINRw.d informat 470
NLMNIJPYw.d format 195
NLMNIJPYw.d informat 471
NLMNIKKRw.d format 196
NLMNIKKRw.d informat 472
NLMNITLw.d format 197
NLMNITLw.d informat 473
NLMNILVLw.d format 198
NLMNILVLw.d informat 474
NLMNIMOPw.d format 199
NLMNIMOPw.d informat 475
NLMNIMXNw.d format 200
NLMNIMXNw.d informat 476
NLMNIMYRw.d format 201
NLMNIMYRw.d informat 477
NLMNINOKw.d format 202
NLMNINOKw.d informat 478
NLMNINZDw.d format 203
NLMNINZDw.d informat 479
NLMNIPLNw.d format 204
NLMNIPLNw.d informat 480
NLMNIRUBw.d format 205
NLMNIRUBw.d informat 481
NLMNISEKw.d format 206
NLMNISEKw.d informat 482
NLMNISGDw.d format 207
NLMNISGDw.d informat 483
NLMNITHBw.d format 208
NLMNITHBw.d informat 484
NLMNITRYw.d format 209
NLMNITRYw.d informat 485
NLMNITWDw.d format 210
NLMNITWDw.d informat 486
NLMNISUDw.d format 211
NLMNISUDw.d informat 487
NLMNIZARw.d format 212
NLMNIZARw.d informat 488
NLMNLAEw.d format 213
NLMNLAEw.d informat 489
NLMNLADw.d format 214
NLMNLADw.d informat 490
NLMNLGBNw.d format 215
NLMNLGBNw.d informat 491
NLMNLGBRLw.d format 216
NLMNLGBRLw.d informat 492
NLMNLCAw.d format 217
NLMNLCAw.d informat 493
NLMNLCw.d format 218
NLMNLCw.d informat 494
NLMNLCNYw.d format 219
NLMNLCNYw.d informat 495
NLMNLCKw.d format 220
NLMNLCKw.d informat 496
NLMNLDKw.d format 221
NLMNLDKw.d informat 497
NLMNLLEKw.d format 222
NLMNLLEKw.d informat 498
NLMNLEGw.d format 223
NLMNLEGw.d informat 499
NLMNLEURw.d format 224
NLMNLEURw.d informat 500
NLMNGBPw.d format 225
NLMNGBPw.d informat 501
NLMNLHKDw.d format 226
NLMNLHKDw.d informat 502
NLMNLHRKw.d format 227
NLMNLHRKw.d informat 503
NLMNLHUFw.d format 228
NLMNLHUFw.d informat 504
NLMNLIDRw.d format 229
NLMNLIDRw.d informat 505
NLMNLISw.d format 230
NLMNLISw.d informat 506
PROC DBCSTAB statement 650
PROC TRANTAB statement 674
PS= system option 600
Puerto Rico
  monetary format 211, 248

R
region codes
  current two-letter code 341
remote applications
  illegal data 601
RENCODING option 638
REPLACE statement 676
  TRANTAB procedure 677
right-alignment
  character expressions 361
RSASIOTRANSERROR system option 601
Russia
  monetary format 205, 242

S
SAS data sets
  naming 607
SAS data views
  naming 607
SAS language elements
  using encoding values 713
SAS sessions
  default character-set encoding 587
  default encoding 22
  encoding 21
  encoding behavior in 25
  locale of 591
  setting encoding of 23
  specifying locale during 7
  system options for DBCS 711
SAS string functions
  internationalization compatibility for 307
SAS/CONNECT
  Compute Services 31
  Data Transfer Services 31
  Remote Library Services 31
SAS/GRAPH
  translation tables in 672
SAS/SHARE
  Remote Library Services 31
  SASMSG function 387
  SASMSGL function 390
  SAVE statement 663, 677
  TRANTAB procedure 678
SBCS encoding 12
  encoding values for transcoding data 713
  searching
    for a word, by position in a string 568
    for specific characters in character expression 350
    for string of characters in character expression 347
  segments
    storing long macro variable values in 571
  SETLOCALE function 395
  shift out/shift in (SO/SI) 39
    trimming from character expressions 370
  shift-code data
    adding to DBCS data 121, 441
    removing from DBCS data 122, 440
Singapore
  monetary format 207, 244
  single-byte character set (SBCS) 12
  encoding values for transcoding data 713
Slovenia
  monetary format 187, 224
SO/SI (shift out/shift in) 39
  trimming from character expressions 370
  sort keys
    linguistic 392
  SORT option 617
  SORT procedure
    language-specific collating sequence for 602
    translation tables in 671
  sorting
    translation tables for 688
  SORTKEY function 392
  SORTSEQ= system option 602
South Africa
  monetary format 212, 249
South Korea
  monetary format 196, 233
Spain
  monetary format 187, 224
  split character strings 40
SQL procedure
  language-specific collating sequence for 602
  standards organizations 15
  string functions
    internationalization compatibility for 307
  strings
    length of 567
    locating first character in 566
reducing length of 568
searching for words by position in 568
substring of a character string 570
substrings
extracting from an argument 366
extracting from an argument, based on
byte position 367
of a character string 570
SWAP statement 678
TRANTAB procedure 672
Sweden
monetary format 206, 243
Switzerland
monetary format 181, 218
system options
  DBCS values for 711
  for transcoding 30
  in SAS sessions for DBCS 711

T
TABLE= option
  SAVE statement (TRANTAB) 678
Taiwan
  monetary format 210, 247
Taiwanese dates 126, 444
Thailand
  monetary format 208, 245
time 5
time values
  converting to specified locale 386
  Time Zone 41
  TIMESZONE= system option 604
trailing blanks
  trimming 52
transcode attributes
  of data set variables 417
TRANSCODE= Column Modifier on
  PROC SQL 637
TRANSCODE= option
  ATTRIB statement 641
transcoding 12, 27
  between operating environments 30
  by specified translation table 412
  compatible and incompatible encodings 32
  considerations for 31
  EBCDIC and OpenEdition encodings 33
  enabled for specified argument 420
  enabled for specified character variable 419
  encoding values for 713
  line-feed characters 32
  preventing 33
  reasons for 28
  SAS options for 30
  suppressing 59
  transferring 59
  versus encoding 12
transcoding errors 601
translating character expressions 368
translation tables 19, 669
  applying to transport files 645
  character sets and 670
  clearing positions 674
  creating 679
  definition 669
  editing 682, 684, 690
  hexadecimal representation of 675
  in SAS/GRAPH 672
  in SORT procedure 671
  inverse tables 686
  loading into memory for editing 676
  modifying 671
  outside TRANTAB procedure 671
  positions 669
  replacing characters in 677
  saving 678
  sorting data 688
  specifying 605
  storing 670
  swapping 672
  transcoding and 28
  transcoding by specified table 412
  transport files
  applying translation tables to 645
  TRANTAB function 412
  TRANTAB procedure 669, 673, 0
  CLEAR statement 674
  concepts 670
  creating translation tables 679
  inverse translation tables 686
  LIST statement 675
  LOAD statement 676
  modifying translation tables 671, 682,
  684, 690
  PROC TRANTAB statement 674
  REPLACE statement 677
  SAVE statement 678
  storing translation tables 670
  SWAP statement 672
  translation tables and character sets 670
  translation tables for sorting 688
  TRANTAB statement
    UPLOAD procedure 643
    TRANTAB-ODS option 636, 642
    TRANTAB= system option 605
    trimming trailing blanks 52
    truncating numeric values 371
Turkey
monetary format 209, 246
TWO option
  CLEAR statement (TRANTAB) 674
  LIST statement (TRANTAB) 675
  SAVE statement (TRANTAB) 678
TYPE= option, TRANTAB statement 643
TZ= system option 604
TZID
  TZONEID Function 402
  TZONEDSTOFF function 408
  TZONEID function 402
  TZONENAME function 404
  TZONEOFF function 405
  TZONE2U function 406
  TZONESTTNAME function 409
  TZONESTTOFF function 410
  TZONEU2S function 411

U
Unicode 14
  BOM prefix on external files 579
  encoding values for transcoding data
    713
    length of character unit 416
    length of display unit 417
  Unicode Consortium 15
  UNICODE function 413
  UNICODEC function 414
  UNICODELEN function 415, 416
  UNICODEWIDTH function 416, 417
United Arab Emirates
  monetary format 176, 213
United Kingdom
  monetary format 188, 225
United States
  monetary format 211, 248
UNIX
  encoding values 723
UPLOAD procedure
  TRANTAB statement 643
uploading files
  translation tables 643
uppercase characters
  changing to lowercase 51
  converting arguments to 372
URLENCODING= system option 612
UTF-16 14
UTF-32 15
UTF-8 14

VALIDMEMNAME system option 607
VALIDVARNNAME= system option 610
variable names
  rules for valid names 610
variables
  associating formats with 640
  associating informats with 640
  labels 640
  length, associating with 640
  transcoding enabled for specified
    character variable 419
  variant characters 16
VARTRANSCODE function 417
VERBOSE option
  PROC DBCSTAB statement 650
VERIFY option
  PROC DBCSTAB statement 650
visual server
  storing logical-ordered text on 123, 124
VTRANSCODE function 419
VTRANSCODEX function 420

W
WEEKUw. format 293
WEEKVw. format 294
WEEKWw. format 296
Windows
  encoding values 724
  ISO encodings 13
  Latin1 code page 9
  words
    searching for, by position in a string 568

X
XMLENENCODING option 643

Y
yen signs, removing 560
YENw.d format 297
YENw.d informat 560
YYWEEKUw. format 298
YYWEEKVw. format 300
YYWEEKWw. format 301

Z
z/OS
  encoding values 725
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.